НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ В.П. Тарасенко
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050102 «Комп'ютерна інженерія»

на тему: «Програмний комплекс для моделювання роботи підприємств агропромислового комплексу»

Виконав:
студент IV курсу, групи КВ-53
Ждан О.О. _____

Керівник:
Доцент кафедри СПСКС, к.т.н.,
Клятченко Я.М _____

Консультант з нормоконтролю:
Доцент кафедри СПСКС, к.т.н.,
Клятченко Я.М _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050102 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Ждану Олегу Олеговичу

1. Тема проекту «Програмний комплекс для моделювання роботи підприємств агропромислового комплексу», керівник проекту Клятченко Ярослав Михайлович, старший викладач, затверджені наказом по університету від «22»травня 2019 р. №1330-С

2. Термін подання студентом проекту «16» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - Аналіз існуючих рішень моделювання агропромислових підприємств та обґрунтування теми дипломного проекту;
   - Особливості програмного моделювання агропромислового елеваторного комплексу підприємств;
   - Аналіз технології розробки імітаційної моделі елеваторного підприємства та обґрунтування їх використання;
   - Особливості програмного комплексу для моделювання елеваторного підприємства аналіз розробленого графічного інтерфейсу.

5. Перелік обов'язкового графічного матеріалу:
   - Схема програмного взаємозв'язку елементів локального сховища даних. Схема структурна.

- Алгоритм відвантаження зернової культури з одного елемента в інший. Блок-схема.
- Структура взаємодії компонентів React і контролерів стану. Схема структурна.
- Алгоритм заповнення матриці суміжності для пошуку шляхів між активними елементами. Блок-схема.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М.., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	17.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Аналіз існуючих рішень	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Розроблення програмного забезпечення	03.02.2019	
6.	Відлагодження програмного продукту	10.02.2019	
7.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
8.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
9.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
10.	Підготовка графічної частини дипломного проекту	19.05.2019	
11.	Оформлення документації дипломного проекту	26.05.2019	

Студент О.О Ждан

Керівник проекту Я.М. Клятченко

АНОТАЦІЯ

Об’єкт розробки – створення релевантної імітаційної моделі агропромислового елеваторного підприємства, математична модель якої заснована на реальних фінансових даних.

Розроблений програмний комплекс – є веб орієнтованим клієнтським односторінковим додатком (Single Page Application) створений за допомогою відкритих javascript бібліотек ReactJs та MobxJS. Серверна частина додатку релізована мовою NodeJs з використанням NoSql баз даних MongoDB.

Створена імітаційна модель, дозволяє моделювати налаштування сховища зерна, оцінку вартості будівлі інфраструктури елеваторного підприємства, експлуатацію, логістичну маршрутизацію і припускати середньостатистичний річний коефіцієнт повернення капіталу на заданих налаштуваннях елеваторного підприємства, а також зберігати стан вже створених моделей елеватора.

В ході розробки дипломного проекту:

- проведено аналіз методів побудови моделі агропромислових підприємств;
- створено математичну модель елеваторного підприємства;
- розроблено програмний комплекс для моделювання елеваторного підприємства;
- Розроблено серверну частину додатка, для збереження стану створеного елеваторного підприємства;
- Розроблений користувацький інтерфейс;

Використання даного програмного комплексу дозволяє перевіряти бізнес-плани і попередньо тестувати різні управлінські рішення в різних умовах, оцінювати навички співробітників і аналізувати результати кожної дії, яку потрібно виконати для поліпшення прийняття рішень в реальному бізнесі.

Ключові слова:

ReactJS, MobX, NodeJs, NoSql, MongoDb, Single page application, Minimum viable product, web-services, simulation modeling.

ANOTATION

Object of development - creation of a relevant simulation model of agro-industrial elevator enterprise, the mathematical model of which is based on real financial data.

The developed software package is a web-based single-page application created with the help of open javascript libraries ReactJs and MobxJS. The server part of the application is released in NodeJs using NoSql MongoDB Databases.

A simulation model is created that allows you to model the grain storage settings, estimate the cost of the infrastructure of the elevator enterprise infrastructure, operation, logistics routing, and assume the average annual return on return on a given elevator setting, as well as maintain the state of the already established elevator models.

During development of the diploma project:

- an analysis of methods for constructing models of agricultural enterprises;;
- a mathematical model of elevator enterprise was created;
- developed a software package for modeling elevator company;
- developed backend application to save state established elevator company;
- designed User Interface;

Using this software system allows you to check business plans and pre-test different management solutions in different conditions, evaluate employee skills and analyze the results of each action that needs to be performed to improve decision-making in real business.

Keywords:

ReactJS, MobX, NodeJs, NoSql, MongoDb, Single page application, Minimum viable product, web-services, simulation modeling.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.454200.002 ТЗ	Програмний комплекс	4		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Технічне завдання			
	А4	ІАЛЦ.454200.003 ТП	Програмний комплекс	2		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ. 454200.004 ПЗ	Програмний комплекс	50		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.454200.001 ОА		
Розробив		Ждан О.О			Програмний комплекс для моделювання роботи підприємств агропромислового комплексу ***Опис альбому***	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Клятченко Я.М.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПМ, КВ-53	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.454200 .005 Д1	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Схема структурна			
	А4	ІАЛЦ.454200.006 Д2	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Блок схема			
	А4	ІАЛЦ.454200 .007 Д3	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Схема структурна			
	А4	ІАЛЦ.454200 .008 Д4	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Блок схема			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.454200.001 ОА	Арк. 3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмна разробка для моделювання агропромислового комплексу підприємств».

Галузь застосування: перевірка бізнес планів та аналіз різних сценаріїв агропромислових підприємств, навчання персоналу агропромислових комплексів та оцінка вміння прийняття рішень працівників агропромислових підприємств.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи ступеня «бакалавр комп'ютерної інженерії», затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення релевантної імітаційної моделі агропромислового елеваторного підприємства, математична модель якої заснована на реальних фінансових даних.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою, що має встановлений NodeJs, NPM (Node Package Manager) або Yarn
- Можливість авторизації користувача та збереження користувацьких данних.
- Наявність інтерфейсу швидкого налаштування вхідних данних симуляції.
- Відтворенні поведінки досліджуваної системи (Елеваторного підприємства) на основі результатів аналізу найбільш суттєвих взаємозв'язків між її елементами.
- Реалізувати можливість перегляду результатів впливу користовача на змодельованну систему за допомогою відповідних елементів користувацького інтерфейсу.
- генерація проекту, побудованого згідно останніх найкращих практик та бібліотек, фреймворків;
- можливість компіляції та запуску згенерованого веб-сервісу;
- Наявність користувацького інтерфейсу.

5.2. Вимоги до апаратного забезпечення

- Процесор: 2-х ядерний, Intel, AMD;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Unix-подібні системи;
- Наявність встановленного GoogleChrome або іншого браузеру.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 454200.004 ПЗ	Програмний комплекс	50		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Пояснювальна записка			
	А4	ІАЛЦ.454200 .005 Д1	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Схема структурна			
	А4	ІАЛЦ.454200.006 Д2	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Блок схема			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ. 454200.003 ТП		
Розроб.		Ждан О.О.			Програмний комплекс для моделювання роботи підприємств агропромислового комплексу *Відомість технічного проекту*	Літ.	Лист 1 / Листів 2
Перев.		Клятченко Я.М.					
Н. контр.		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-53	
Затв.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.454200 .007 Д3	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Схема структурна			
	А4	ІАЛЦ.454200 .008 Д4	Програмний комплекс	1		
			для моделювання роботи			
			підприємств			
			агропромислового			
			комплексу			
			Блок схема			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.454200.001 ОА	Арк. 2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

JavaScript – мова програмування високого рівня.

ReactJs – відкрита бібліотека JavaScript для створення односторінкових веб додатків, суть якої полягає в роботі з Shadow Dom та швидким оновленням DOM дерев на стороні клієнта(SPA)

Shadow DOM – це частина документа, що реалізує інкапсуляцию в DOM дереві.

Document Object Model (DOM) - специфікація програмного інтерфейсу для роботи зі структурованими документами.(XML, HTML та ін.)

Single page application – один з методів розробки веб додатків, сутність якого полягає в рендері веб додатку на клієнтській стороні з допомогою середовища розробки JavaScript

MobX – відкрита JavaScript бібліотека, що поліпшує контролювання стану класів для односторінкових веб додатків.

NodeJs - програмна платформа, що здійснює трансляцію JavaScript в машинний код, та перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення.

MongoDB – документо-орієнтована база даних, яка класифікується як NoSQL база даних.

Webpack – відкрита програмна бібліотека, основною метою якої є поєднання JavaScript-файлів для використання в браузері, але він також здатний перетворювати, комплектувати або упаковувати відокремлені модулі javascript, якщо на це є необхідність.

Scalable Vector Graphics (SVG)– XML орієнтоване векторне зображення, що може керуватися методами JavaScript.

Cascading Style Sheets (CSS) - це мова таблиць стилів, використовуваний для опису представлення документа, написаного на HTML

Hypertext Markup Language (HTML) – мова розмітки для створення веб додатків, яка зрозуміла браузеру.

Enterprise Resource Planning System (ERPS) - Система планування ресурсів підприємства

ПЗ – програмне забезпечення.

БД – база даних;

ООП – Об'єктно-орієнтоване програмування.

ВСТУП

Навіть якщо порівнювати з минулим десятиліттям, в сучасному світі технології зробили крок далеко вперед. Агропромислові підприємства з кожним роком збільшують свої виробничі потужності за рахунок впровадження нових технологій в методи виробництва аграрної продукції. Однак дуже часто співробітники агропромислових підприємств можуть приймати неправильні рішення, вважаючи що перевиробництво за рахунок впровадження нових технологій у виробничий процес може значно збільшити дохід підприємства, на відміну від належних інвестицій в розвиток самого робочого персоналу та розробки правильної методології прийняття рішень. Так збиток деяких підприємств через некомпетентність робочого персоналу може досягати мільйони умовних одиниць. Будь то помилка механізатора, неправильно розроблена логістична схема перевалочного підприємства або бездумне слідування картками техоперацій не вдаючись у подробиці регіонального розміщення підприємства, ринкових цін і специфіки виробництва - все це негативно вплине на доходи і повернення капіталу підприємства.

Звичайно додаткову складність становить сама дорожнеча агропромислових підприємств, де навчання персоналу всім хитрощам аграрної професії може зайняти значний час, а помилки на реальному підприємстві можуть коштувати надто дорого.

Останнім часом великі підприємства все частіше починають інвестувати гроші в розвиток персоналу і побудова імітаційних моделей аграрного підприємства, які засновані на вже існуючих статистичних даних. Імітаційні моделювання на відміну від звичайного математичного моделювання являють собою не просто зріз даних, а наближення до реальних екосистем аграрних підприємств з можливістю моделювання тривалих і коштовних процесів на коротких проміжках часу і витрачаючи менше ресурсів. Імітаційне

моделювання дозволяє швидко та ефективного моделювати різні сценарії ключових рішень в аграрному виробництві та прогнозувати вплив кожного рішення на очікувані результати та повернення інвестицій.

Завдання дипломного проекту - це побудова моделі елеваторного агропромислового комплексу, яка дозволила б спланувати вартість витрат на будівництво інфраструктури елеватора, обсяги надходження зернових культур, пропускну здатність елеватора при різній комплектації обладнання, а також оцінити витрати і повернення капіталу за один календарний рік функціонування підприємства .

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ МОДЕЛЮВАННЯ АГРОПРОМИСЛОВИХ ПІДПРИЄМСТВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Аналіз існуючих методів моделювання агропромислових підприємств.

У наш час існує багато різноманітних методів прийняття рішень. Один з таких методів - це побудова емуляції реальних процесів системи на умовній моделі з метою отримати на виході набори даних, які змогли б охарактеризувати створену модель.

Потреба в засобах аналізу і опису бізнес-процесів задає необхідність розробки нових методів моделювання і аналізу. Модель відтвореної системи повинна дозволяти коректно описати динаміку і статистику функціонування окремих процесів, а також діяльність всього підприємства в цілому.

В силу багатозначності поняття «модель» - не існує єдиної класифікації видів моделювання, але класифікацію можна проводити по характеру модельованих об'єктів, за характером моделей, та за сферами застосування.

Виділяють такі види моделювання:

- Інформаційне моделювання;
- Математичне моделювання;
- Статистичне моделювання;
- Економіко-математичне моделювання;
- Імітаційне моделювання;
- Фізичне моделювання та інші;

Серед усіх видів можна виділити ряд моделювань, що належать до аналітичних. До таких моделювань буде відноситись інформаційне, математичне, статистичне або економіко-математичне.

Базовим методом моделювання - є Математичне моделювання, і воно є фундаментом для будь-якої іншої моделі. Математичне моделювання ґрунтується на математичних методах подання реальності, за допомогою якої, зазвичай, описується ідеальний об'єкт або процес. Таке моделювання стає базою не тільки для економічних і бізнес систем, а також фізичних, наукових, соціологічних та інших процесів.

Плюси:

- Дозволяє побудувати точну модель, засновану на математичних методах
- Для створення вимагає малу кількість витрат ресурсів

Мінуси:

- Не описує поведінку моделі при виникненні випадкових явищ
- Не показує повної структури взаємозв'язків емульованної моделі, а всього лише зріз вхідних і вихідних даних.

Другий найбільш поширеним методом моделювання економічних і бізнес процесів - є Статистичне моделювання. Статистичне моделювання, на відміну від математичного ґрунтується на статистиці і допомагає враховувати виникнення випадкових подій, які складно врахувати за допомогою математичних моделей, а також дозволяють врахувати кількісну сторону масових явищ в числовій формі. Статистичне моделювання ґрунтується на статистичних методах.

Плюси:

- Враховує випадкові події.
- Допомагає дати кількісну оцінку характеристик побудованої моделі
- Для створення вимагає малу кількість витрат ресурсів

Мінуси:

- Має високу ймовірність похибки
- Вимагає безліч кількісних досліджень

- Не показує повної структури взаємозв'язків емульованної моделі, а всього лише зріз вхідних і вихідних даних.

Наступний вид моделювання - це інформаційне моделювання. Інформаційне моделювання являє досліджуваний об'єкт у вигляді логічної інформації та описує суттєві для даного розгляду параметри і змінні величини об'єкта, зв'язку між ними, які дозволяють шляхом подачі на модель вхідних величин моделювати можливі стани об'єкта. Тобто інформаційне моделювання допомагає нам змоделювати логічну модель підприємства і скласти схему його функціонування.

Плюси:

- Показує логічні взаємозв'язки між елементами системи.
- Дозволяють охарактеризувати змінні величини об'єкта
- Для створення вимагає малу кількість витрат ресурсів

Мінуси:

- Описує тільки взаэмозв’язок елементів системи.

Всі перераховані вище методи активно використовуються на агропромислових підприємства, але мають ряд істотних недоліків, оскільки математична, статистична або логічна моделі, які відокремлені одна від одної, мають високий поріг входу для розуміння побудованої моделі, а також вони вкрай обмежені та статичні, за рахунок вузького аналізу у своїй предметній області, що не дає розуміння всіх можливих аспектів функціонування підприємства. Також моделювання нового об'єкта навіть з несуттєво відмінними параметрами, представленими в логічній схемі підприємства - змушують вносити в існуючі моделі ряд складних і трудомістких модифікацій.

На основі цих аналітичних моделей прийняття деяких неочевидних, але правильних рішень стає просто неможливо, так як вони представляють собою виключно зріз вхідних і вихідних даних і спрогнозувати результати тих чи інших дій в довгостроковій перспективі дуже складно.

Серед програмних розробок, що використовують перераховані вище способи моделювання найбільш часто зустрічаються різноманітні система планування ресурсів підприємства (ERPS). ERPS – зазвичай корпоративна інформаційна система, призначена для автоматизації обліку й управління, що допомагають управляти фінансами, формуванням та розподілом запасів, реалізацією та маркетингом, утриманням покупців тощо. Серед недоліків ERP систем э те, що невеликі компанії не можуть дозволити собі інвестувати достатньо грошей в ERP і адекватно навчити всіх співробітників, а помилки розробників у системі приводять до відчутних втрат коштів та долі на ринку.

Один з видів моделювання, який не відноситься до аналітичних видам моделювання - імітаційне моделювання. Існує клас об'єктів, для яких з різних причин не розроблені аналітичні моделі, або створення аналітичної моделі принципово неможливо, або не розроблені методи рішення моделі, або рішення нестійкі. В цьому випадку аналітична модель замінюється імітаційної моделлю.

На відміну від аналітичних моделей - імітаційне моделювання більш гнучке і масштабуємо, а також дозволяє використовувати його для навчання персоналу і більш простому виявленню причинно-наслідкових зв'язків, тому що імітаційна модель є інтерактивним об'єктом і дозволяє користувачеві втручатися в процес симуляції.

Імітаційне моделювання вирішує такі проблеми:

- неможливо побудувати аналітичну модель: в системі є час, причинні зв'язку, наслідок, не лінійності, стохастичні (випадкові) змінні;
- дорого або неможливо експериментувати на реальному об'єкті;
- необхідно зімітувати поведінку системи в часі;

1.2. Обґрунтування теми дипломного проекту

Прийняття рішень - складова частина будь-якої управлінської функції. Необхідність прийняття рішень виникає на всіх етапах процесу управління, пов'язана з усіма ділянками та аспектами управлінської діяльності.

Імітаційна модель бізнес процесів аграрного виробництва, заснована на статистичних даних і аналітичних алгоритмах дозволяє швидко і ефективно змоделювати різноманітні сценарії ключових рішень в аграрному виробництві і спрогнозувати їх результати, а також вплив кожного рішення на ефективність і фінансові показники.

Мета дипломної роботи - це побудова інтерактивної імітаційної моделі елеваторного агропромислового комплексу, яка базується на існуючих математичних, логічних і статистичних моделях і дозволяє:

- спланувати вартість витрат на будівництво інфраструктури елеватора;
- спланувати обсяги надходження зернових культур, пропускну здатність елеватора, при різній комплектації обладнання;
- оцінити витрати і повернення капіталу за один календарний рік функціонування підприємства;
- Створити симуляцію робочих процесів та випадкових подій, що відбуваються на елеваторному підприємстві.

Створена програмна розробка дозволить:

- перевіряти бізнес плани та аналізувати різні сценарії, що дозволяють уникати деяких ризиків при інвестуванні та обранні оптимальних напрямки рішень в реальних сценаріях виробництва;

- Навчати персонал елеваторних підприємств всіх рівнів та створення більш детального розуміння бізнес процесів у людей, опосередковано пов'язаних з виробництвом;
- Оцінити персонал для виявлення прогресивних учасників підприємства, а також виявити не тільки рівень професійних навичок, але і відношення людини до роботи та звички у прийнятті рішень;

2. ОСОБЛИВОСТІ ПРОГРАМНОГО МОДЕЛЮВАННЯ АГРОПРОМИСЛОВОГО ЕЛЕВАТОРНОГО КОМПЛЕКСУ ПІДПРИЄМСТВ

2.1. Особливості елеваторного підприємства

Зерновий елеватор – це аграрний комплекс, призначений для накопичення або зберігання зернових культур і доведення їх в стан, який задовольняє експортним нормам, тобто базисним показниками засміченості і вологості зернових культур.

Серед усіх елементів зернового елеватору ми виділимо такі основні елементи, необхідні для справної роботи підприємства:

- Вагова - елемент обладнання перед приймальним відділенням який складається з автомобільних і залізничних ваг для зважування зернових культур;
- Приймальне відділення - являє собою комплекс завальних ям різних об'ємів для вивантаження залізничного або автотранспорту в яку зсипається зернові культури для подальшого їх надходження на конвеєр;
- Транспортер і норії - елемент обладнання елеватора призначений для транспортування і підйому зернових культур між різними елементами елеваторного підприємства починаючи з завальних ям, закінчуючи авто або жд відвантаженням.
- Чистячі споруди - елемент обладнання елеватора, який складається з машини для попередньої, первинної і, при необхідності, вторинної очистки зерна і аспірації від легких домішків з подальшим їх доведенням до базисних показниками засміченості.
- Сушильне відділення - елемент обладнання елеватора, який включає в себе ємності для накопичення вологого і сухого

матеріалу, а також сушарки які використовуються для первинної або вторинної обробки зернових культур з подальшим їх доведенням до базисних показником вологості.

- Силосний корпус - складається з силосів і є основним елементом зберігання елеватора.
- Хопери або конусні силоси - елементи зберігання малого обсягу зернових культур, які використовуються як резервуар для зберігання, перед попереднім відвантаженням або чищенням/сушінням зерна.
- Відвантаження - станція жд або авто відвантаження зернових культур.
- Адміністративно-побутовий корпус, лабораторія, пожежний резервуар і інші, необхідні за нормативами, будівлі та споруди.

Залежно від призначення елеватори поділяють на:

- хлібоприймальне або заготівельні - ємність 20-100 тис. т приймають зерно від господарств, очищають від домішок, сушать і відвантажують споживачеві;
- виробничі - ємність 10-150 тис. т, споруджують при круп'яних, крохмалепатокових, комбікормових заводах, млинах і т.д.;
- базисні – ємність 100-150 тис. т, призначені для тривалого зберігання зерна, прийнятого з ж / д транспорту і відвантажується на жд
- перевалочні і портові - ємність 50-100 тис. т, будують в місцях перевалок зерна з одного виду транспорту на інший - в морських портах, на ж/д станціях і т.д.

2.2. Виявлення основної проблематики створення елеваторного підприємства.

Елеваторне підприємство - це дороге спорудження, вартість якого іноді досягає десятків мільйонів у.о., особливо це стосується елеваторів базисного і перевалочного типу.

Потік зернових культур на таких елеваторах, іноді може перевищувати обсяг ємностей зберігання підприємства в 2 або в 3 рази, що передбачає наявність у керівників елеваторного підприємства належної вправності і умінь приймати рішення в галузі логістики внутрішніх процесів підприємства.

Створення таких елеваторів гігантів змушують враховувати регіональні особливості розподілу культур, пропускну здатність елеваторного підприємства, ретельно підбирати необхідні типи обладнання під самі затребувані види зернових культур.

Побудова самих новітніх елеваторів зазвичай займає від одного до 3х років. Кожен елеватор в регіоні має загальний тариф на обслуговування, зберігання, перевалку, створення документації і приведення до базисних показників засміченості і вологості зернових культур.

Розрахунковий час використання більшості елементів обладнання елеватору – 15 років. Статистика показує, що найновіші, добре обладнані і продумані елеватори мають в середньому близько 20 - 27% повернення капіталу в рік, а це значить, що ідеальне підприємство з урахуванням часу побудови - почне приносити прибуток як мінімум через 5 - 7 років. Але якщо виробництво не було оптимізовано, то воно може почати приносити дохід через 10 або більше років, або не приносить його зовсім. Саме з цієї причини продумування всіх елементів елеваторного підприємства, облік регіональних особливостей і вимог ринку, якісне навчання персоналу і створення правильної методології прийняття рішень і скорочення часу окупності

елеватора - може зекономити мільйони і підвищити продуктивність підприємства в 2 або 3 рази.

2.3. Побудова аналітичної моделі елеватора**.**

Статистичні дані для моделювання елеваторів були отримані на підставах звітів елеваторів Агродар Бар, Житомирського регіонального елеватора і ряду інших елеваторних підприємств у Вінницькій, Миколаївській та Сумській області компанії "Агропросперіс" за період 2016-2018 р, а також інформаційних ресурсів компанії “LatifundistGroup”.

Для побудови моделі елеватора ми будемо використовувати дані найбільш розповсюджені зернові культур в Україні:

- Ріпак
- Пшениця
- Соняшник
- Соя
- Кукурудза

Перш за все варто зазначити, що місткість всіх елементів зберігання і пропускна здатність окремо взятих елементів вимірюється в тоннах. Однак при розрахунку обсягу вважається що ємність буде заповнювати матеріал, щільність якого становить 1000 кілограм на метр кубічний. Звичайно, зернові культури мають меншу щільність, а тому для отримання реального обсягу для ємності зберігання або пропускної здатності кожного елемента потрібно їх помножити на коефіцієнт об'ємної ваги для окремої культури. (таблиця 3.1)

Таблиця 3.1

Коефіцієнт об'ємної ваги для кожної з культур

Ріпак	Пшениця	Соя	Кукурудза	Соняшник
0.8	0.96	0.86	0.88	0.52

Тобто для повністю заповненого соняшником силосного сховища номіналом в 5 000т, реальна вага зернової маси буде 2600т. Отже для отримання повністю заповненого елементу зберігання треба використати формулу 2.3.1:

$$M_c = kM \quad (2.3.1.)$$

Де M_c – реальна вага, k – коефіцієнт об'ємної ваги, M - ємність елементу зберігання.

Для отримання пропускної здатності окремо взятого елемента треба скористатися формулою 2.3.2:

$$V_c = kV \quad (2.3.2.)$$

Де V_c – реальна пропускної здатності, k – коефіцієнт об'ємної ваги, V - пропускної здатність елементу зберігання.

На пропускну здатність елеватора впливають такі чинники:

- Швидкість конвеєрних стрічок і норій;
- Кількість прийомних відділень;
- Кількість і швидкість роботи сушильних та очисних споруд;
- І кількість відвантажувальних станцій.

В якості транспортного обладнання зазвичай використовуються ланцюгові скребкові транспортери, стрічкові конвеєри і норії. Ланцюгові транспортери і стрічкові конвеєри використовуються для горизонтального переміщення зернових культур, а норії для вертикального підйому до елементів зберігання. Тому пропускну здатність транспортного обладнання ми будемо розраховувати з розрахунку типових транспортних конвеєрів.

На елеваторі може знаходиться майже необмежена кількість транспортних конвеєрів зі складною структурою. Зазвичай швидкість транспортного конвеєра складає не більш 0.5 м за секунду, зернова насип не більше 0.10 м заввишки, а довжина конвеєра досягати 2м.

З цих показників ми можемо порахувати пропускну здатність транспортного конвеєру в 1 год за формулою 2.3.3.:

$$V = lhV * 3600c = 360 м^3 / г \quad (2.3.3.)$$

Де V – пропускна спроможність транспортної лінії, l – ширина конвеєрної стрічки, h - висота зернової насипи. Для розрахунку пропускної спроможності на транспортному обладнані для кожної з культур можем використати формулу 2.3.2.

Оптимальний режим сушіння - це створення і підтримка таких умов роботи, при яких забезпечується максимальна продуктивність, поліпшення якості зерна або повне збереження і високі показники роботи сушарки. Основними параметрами режиму сушіння є:

- температура теплоносія, що подається в сушильну камеру;
- температура максимального нагріву зерна в процесі сушіння;
- температура охолодженого зерна;
- час перебування зерна в нагрітому стані, що визначається відсотком знімання вологи;

Очисні споруди для очищення зерна в елеваторі передбачають попереднє очищення від великих і дрібних домішок і виділення дрібної фракції в сепараторі. Знімання відсотка засміченості у зерна під час проходження етапу очисних споруд залежить від часу знаходження самого зерна в механізмі чищення.

Як очисні споруди, так і споруди для сушіння зерна є потоковими елементами час проходження яких залежить від необхідного відсотка зняття засміченості і вологості зернової культуру. Середні показники пропускної здатності сушіння і чищення при різних показниках знімання відсотка вологості і засміченості можна дізнатися в документації обладнання. Приклади обладнання показані в таблиці 3.2 і 3.3 відповідно.

Таблиця 3.2

Приклад показників ефективності обладнання елементів очисних споруд

Назва		очищення (%) Пшениця	очищення (%) Ріпак	очищення (%) Соя	очищення (%) Соняшник	очищення (%) Кукурудза
Buhler SMA-203	%	2-3	2-3	2-3	2-3	2-3
	ТОНН/час	110	60	100	55	80
	%	4-6	4-6	4-6	4-6	4-6
	ТОНН/час	85	55	85	40	68
	%	7-9	7-9	7-9	7-9	7-9
	ТОНН/час	75	45	75	30	55
Buhler TAS 206	%	2-3	2-3	2-3	2-3	2-3
	ТОНН/час	170	140	150	90	150
	%	4-6	4-6	4-6	4-6	4-6
	ТОНН/час	170	100-140	150	80	150
	%	7-9	7-9	7-9	7-9	7-9
	ТОНН/час	130	20-100	110	30-70	150

Таблиця 3.3

Приклад показників ефективності обладнання елементів сушильного віділення

Назва		сушка (%) Wheat	сушка (%) Rape	сушка (%) Soy	сушка (%) Sunflower	сушка (%) Corn
TORNUM HR 8-27-3	%	2-4	2-4	2-4	2-4	2-4
	ТОНН/час	54	30	29	21	42
	%	5-7	5-7	5-7	5-7	5-7
	ТОНН/час	41	21	25	17	38
	%	8-10	8-10	8-10	8-10	8-10
	ТОНН/час	33	17	17	13	34
	%					11-13
	ТОНН/час					25
	%					14-16
	ТОНН/час					21
	%					17-19
	ТОНН/час					17

З даних по пропускній здатності устаткування сушіння і чищення, можемо помітити, що залежність зняття засміченості і вологості зерна від часу знаходження в елементі обробці зернових культур має парабалічну залежність. Тому для спрощення моделі пропускну здатність сушки і чищення, при

чищенні і сушці зернових культур до базових показників, які відповідають нормі - будемо вираховувати автоматично базуючись на показниках засміченості і вологості за формулою 2.3.4.

$$V = \begin{cases} V_{\min}, W > S_{crit} \\ V_{\max}, W < S_{base} \\ \dfrac{(W - S_{crit})^2 * (V_{\max} - V_{\min})}{(S_{crit} - S_{base})^2} + V_{\min}, S_{base} < W < S_{crit} \end{cases} \quad (2.3.4)$$

Де V_{min} - мінімальна пропускна здатність, V_{max} - максимальна пропускна здатність, S_{crit} – критичні показники засміченості та вологості культури, S_{base} – базові показники засміченості та вологості культури, W - поточний рівень засміченості та вологості культури, V – оптимальна швидкість проходження етапу обробки зерна в тонах на годину. Для розрахунку пропускної спроможності на очисних спорудах для кожної з культур можем використати формулу 2.3.2.

Базові показники засміченості та вологості можна дізнатися у таблицях 3.4 та 3.5 відповідно.

Таблиця 3.4

Базові показники засміченості %

Ріпак	Пшениця	Соя	Кукурудза	Соняшник
3	2	3	2	3

Таблиця 3.5

Базові показники вологості %

Ріпак	Пшениця	Соя	Кукурудза	Соняшник
8	14	12	14	8

Критичні показники засміченості та вологості можна дізнатися у таблицях 3.6 та 3.7 відповідно.

Таблиця 3.6

Критичні показники засміченості %

Ріпак	Пшениця	Соя	Кукурудза	Соняшник
10	5	10	10	10

Таблиця 3.7

Критичні показники вологості %

Ріпак	Пшениця	Соя	Кукурудза	Соняшник
18	20	18	36	16

На приймальному відділенні, якщо зерно надходить стабільної якості, то час операцій з приймання: відбір зразків, проведення аналізу, зважування брутто, вивантаження і виїзд автомобіля з території займає не більше 10-15 хв.

Для моделі елеватора ми робимо спрощення і будемо вважати що кожна вантажівка з зерновою культурою важить 20т без врахування объємного коефіцієнта, а час прийому однієї вантажівки 15 хв. Отже нескладно підрахувати, що пропускна здатність одного прийомного відділення 80т об’ємної ваги за годину. Для розрахунку пропускної спроможності на прийомному відділені для кожної з культур можем використати формулу 2.3.2.

Пропускну спроможність на станції відвантаження ми враховуємо так само як і на приймальному відділені 80т/г. Для потягів пропускна здатність буде трохи більше, оскільки потяги мають більший об’єм, який ми умовно приймаємо за 60т в одному вагоні потяга. Важливо зазначити, що на базисних та перевалочних підприємствах усі зернові культури найчастіше відвантажують поїздами, виняток становить кукурудза, яка може

використовуватися для місцевих підприємств і не має особливої цінності, як експортний товар.

Для розрахунку вартості послуг елеватора використовується таблиця 3.8

Таблиця 3.8

Вартість послуг елеватора

		Рапс	Пшениця	Соя	Соняшник	Кукурудза
Приймання	$/тн	0,56	0,56	0,62	1,12	0,62
Очистка	$/тн%	0,5	0,5	0,55	0,71	0,55
Сушка	$/тн%	1,54	1,14	1,69	2,21	1,6
Зберігання	$/тн.мес.	1,67	1,39	1,5	3,05	1,39
Відгрузка	$/тн	3,42	3,14	3,45	3,79	3,45
Документація	$/тн	1,4	1,4	1,4	1,4	1,4

Витрати на елеваторній підприємстві залежать від виконуваних виробничих етапів. Зазвичай у вартість витрат входять фонд оплати праці (ФОП), газ, електроенергія, пально-мастильні матеріали (ПММ), матеріали и прямі послуги. Для непрямих витрат ми враховуємо ФОП, адмін витрати, загальновиробничі витрати і послуги і беремо фіксоване значення витрат в місяць для елеваторного підприємства в залежності від його комплектації. Дані по витратам ми усереднюємо, та для розрахунку прямих витрат елеватора використовуємося таблицю 3.9

Таблиця 3.9

Прямі витрати елеватора

		Рапс	Пшениця	Соя	Соняшник	Кукурудза
Приймання	*$/тн*	*0,31*	*0,31*	*0,31*	*0,33*	*0,31*
Очистка	*$/тн%*	*0,12*	*0,12*	*0,12*	*0,12*	*0,12*
Сушка	$/тн%	0,49	0,48	0,57	0,46	0,57
Зберігання	*$/тн.мес.*	*0,07*	*0,07*	*0,07*	*0,07*	*0,07*
Погрузка	$/тн	0,46	0,46	0,46	0,48	0,46

Для розрахунку повернення капіталу на елеваторній підприємстві за один рік - ми використовуваному співвідношення всіх річних доходів за вирахуванням витрат до загальної вартості елеваторного підприємства.

2.4. Побудова інформаційної моделі елеватору**.**

Описові інформаційні моделі - це моделі, створені на природній мові в усній або письмовій формі. Інформаційна модель елеваторного підприємства (рiс.2.4) відноситься до описової моделі і показує всі можливі взаємозв'язки і переходи між етапами виробництва.

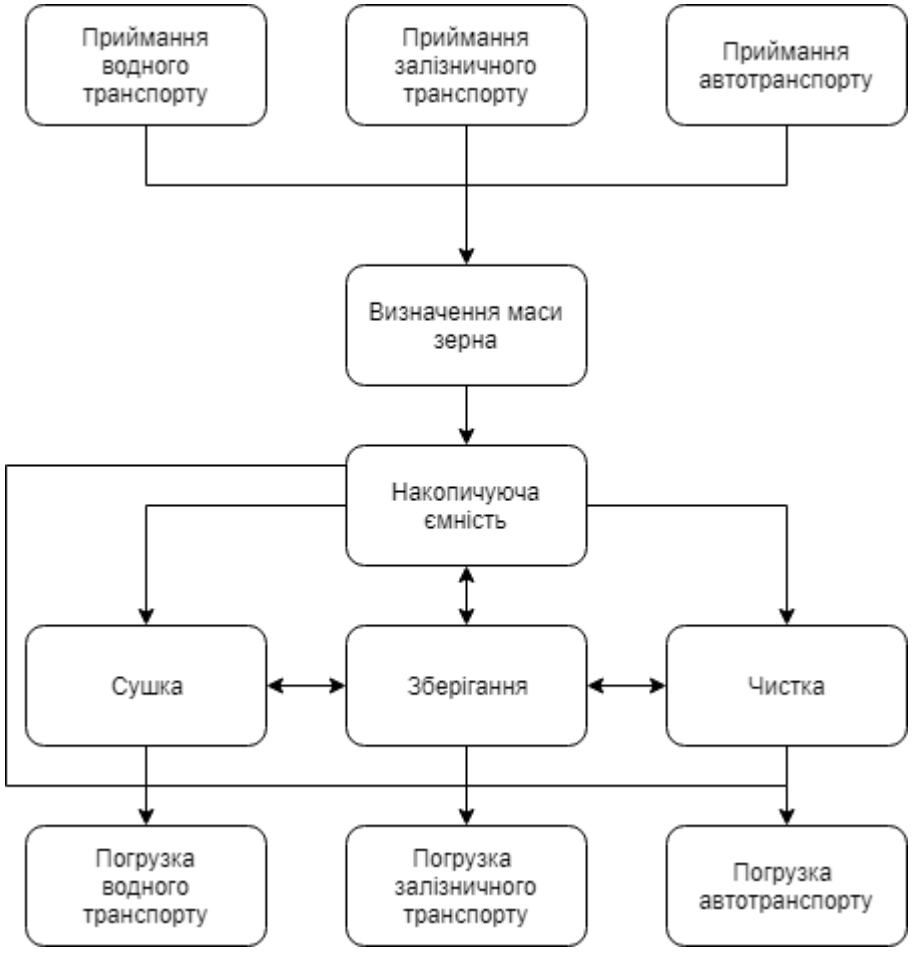


Рисунок 2.4 - Інформаційна модель

3. АНАЛІЗ ТЕХНОЛОГІЇ РОЗРОБКИ ІМІТАЦІЙНОЇ МОДЕЛІ ЕЛЕВАТОРНОГО ПІДПРИЄМСТВА ТА ОБГРУНТУВАННЯ ЇХ ВИКОРИСТАННЯ

При створенні імітаційної моделі передбачається розробка MVP (Minimum viable product) додаток, який можна було б легко перенести на різні платформи з несенням якомога менших витрат реусурсов. MVP - продукт, що володіє мінімальними, але досить наповнений функціонально для задоволення перших споживачів. Основне завдання - отримання зворотного зв'язку для формування гіпотез подальшого розвитку продукту.

Оскільки додаток вимагає мінімального навантаження на сервер і оптимізації трафіку, саме з цього в якості основної технології розробки було вибрано SPA з дуже великим упором на клієнтську частину. Серверна частина необхідна тільки для збереження стану самого додатка і ніяк не бере участь в процесі самої симуляції.

Також можливість перенесоа веб додатка на різні платформи тільки зміцнює вибір технологій для реалізації проекту. Проект надихався такими додатками як MicrosoftStore, Discord або Atlassian, які спочатку були виключно веб орієнтованими програмами, але пізніше переведені на десктопну основу за допомогою фреймворка electron.

Electron - фреймворк, розроблений GitHub заснований на бібліотеки рендеринга Chromium для взаємодії з html.

3.1. Порівняльний аналіз технологій розробки клієнтських односторінкових додатків AngularJs, ReactJs, VueJS.

Angular.js називають MVW (Model-View-Whatever) фреймворк і серед головних вигод називають: швидке написання коду, швидке тестування програми та двостороння прив'язка даних (зміни в бекенді відразу ж відбиваються на призначеному для користувача інтерфейсі).

ReactJs розроблений компанією Facebook і показує свою свою високу ефективність всередині динамічних додатків з великим трафіком. Вважається самим швидким JS фреймвком, хоча фреймворком його можна назвати умовно, оскільки в MVC (Model-View-Controller) у моделі React.js діє як "V" і може бути легко інтегрований в будь-яку архітектуру. Завдяки використанню ShadowDOM дерева він забезпечує зростання продуктивності в порівнянні з Angular 1 На додаток до цього, компоненти React можуть бути створені і повторно використані в інших додатках.

Vue.js пропонує двосторонню прив'язку даних (Сервер-клієнт), і візуалізацію на стороні сервера (як в Angular2 і ReactJS), Vue-cli (інструмент для швидкого старту) і опціональну підтримку JSX. Vue.js кращий вибір для швидкої розробки крос-платформних додатків. Він є вигідним рішенням для тих випадків, коли продуктивність важливіше, ніж хороша організація коду або структура програми.

3.1.1. Порівняльний аналіз рендеру AngularJs, ReactJs, VueJS.

В першу чергу потрібно порівнювати способи та швидкодію рендеру js фреймворків Сучасна архітектура налічує два види: на стороні сервера або на стороні клієнта (за рахунок ПК користувача).

DOM - Document Object Model - об'єктна модель документа, яка дозволяє зчитувати і змінювати вміст, структуру і оформлення html-документів. Кожен з фреймворків підходить по-своєму до обробки DOM що і впливає на рендеринг кінцевої сторінки, яка відображається на екрані користувача

Vue.js і React створюють копію DOM, обробляють її, а потім результат порівнюється з початковою версією. Після порівняльного аналізу в DOM дереві користувача змінюються тільки ті елементи, які відрізняються від початкових елементів. Це значно прискорює рендерінг сторінки, а також скорочується обсяг трафіку, що особливо важливо для користувачів мобільних пристроїв.

У корені відрізняється підхід до обробки DOM фреймворком Angular версії 1.x і вище. Тут відбувається поділ на два потоки. У першому потоці за рендеринг DOM «відповідає» браузер , а другий потік (серверна частина) - відповідає за створення директив, завантаження коду і сервісів. Angular працює значно повільніше порівняно з React та Vue, а також додатково навантажує сервер.

3.1.2. Порівняльний архітектура компонентів.

Як вже було зазначено React не відноситься до фреймворків в чистому вигляді, це модифікована бібліотека, зроблена на прикладі MVC (Model-View-Controller, де Модель відповідає за надання даних, Вид - відображає моделі користувачеві, Контролер інтерпретує дії користувача і вносить зміни в модель).

Angular і Vue.js відносяться до фреймворками. Це значить що angular і Vue ідуть з коробки і не потребують додаткових бібліотек, настройки конфігурації, та встановлення трансляторів мови перед використанням, на відміну від React.

Якщо в основі архітектури проекту лежить React, то це потребує пошуку і впровадження додаткових бібліотек для реалізації певних задач, попереднього налаштування проекту та створення власної архітектури проекта. Також варто зазначити, що такий підхід для кастомізації архітектури проекту може давати значно більшу ефективність порівняно з іншими фреймворками, хоча у разі використання готових фреймворків проблем з підбором або налаштуванням бібліотек для різних завдань вже не виникає.

3.1.3. Направленість та класи залежностей

React і Vue.js підтримують тільки односторонню передачу даних. При цьому в React об'єкти вміщені. Говорячи простою мовою, кожен з об'єктів програми відноситься до кінцевих процедур, які не вимагають дій користувача до закінчення роботи.

Однак React підтримує копіювання і передачу стану. Тобто, властивості прописаних об'єктів можуть бути відновлені на іншому пристрої, якщо запустити додаток і повідомити стан компонентів. Отже, рендеринг буде ідентичним, на екранах обох пристроїв буде одна і та ж «картинка».

Фреймворк Vue.js працює трохи інакше. Він також односторонній, але компоненти працюють з шаблонами, і на виході виходить чистий html. Є підтримка JSX, що спрощує перехід з React і схожих бібліотек.

Фреймворк Angular дещо відрізняється логікою процесу. Так, тут є все, що властиво Vue.js, однак опис взаємодії об'єктів відбувається в службах, які є складовими частинами модулів.

Модульна архітектура більш зручна при розробці великих додатків. Модуль створюється для вирішення декількох завдань схожою функціональності.

Підтримка MVVM (Model-View-ViewModel) допускає рішення різних завдань в одному розділі програми з використанням одного набору даних. Залежність функцій визначає двосторонню спрямованість передачі даних. Кожна процедура може запустити інший процес.

3.2. Обґрунтування вибору технології ReactJs+MobX.

Після проведення порівняльної характеристики фреймворків для створення Single page application. Вибір був зупинений на технолгии ReactJS.

Вибір обгрунтований тим, що React є повністю незалежним від сервера, має декларативний підхід до написання коду і високу швидкодію за рахунок роботи з ShadowDom. Також реактивний додаток не створюється з коробки, що дає нам високу ступінь кастомізації архітектури додатку.

Основна логіка React - це збирання програми з примірників об'єктів, які успадкували компоненти React і підкоряються станам цих компонентів.

При зміні стану компонента, React виявляє невідповідність в DOM дереві, після чого точково їх замінює. В компоненті є дві нативні змінні стану

які називаються props (скорочення від «properties») і state. Кожна з змінних містять інформацію, яка впливає на результат відтворення компонента, але між ними є істотна різниця: props передається компоненту (аналогічно параметрам функції), а state управляється всередині компонента (подібно локальним змінним).

При зміні цих змінних міняється життєвий стан самого компоненту.

При створенні компоненту включаються такі життєві стани:

- constructor() – конструктор, в якому відбувається початкова ініціалізація компонента
- componentWillMount() - викликається перед рендерингом компонента
- render() - рендеринг компоненту
- componentDidMount() - викликається після рендерингу компонента

Також є власні стани при оновленні компоненту:

- shouldComponentUpdate (nextProps, nextState) - викликається кожен раз при оновленні об'єкта props або state. Як параметр передаються новий об'єкт props і state. Ця функція повинна повертати true (треба робити оновлення) або false (ігнорувати оновлення). За замовчуванням повертається true. Але якщо функція буде повертати false, то тим самим ми відключимо оновлення компонента, а наступні функції не будуть спрацьовувати.
- componentWillUpdate(nextProps, nextState) - викликається перед оновленням компонента
- render () - рендеринг компонента
- componentDidUpdate (prevProps, prevState) - викликається відразу після поновлення компонента. Як параметри передаються старі значення об'єктів props і state.

Оскільки додаток, що розробляється досить великий і вимагає наявність якогось локального сховища, то на додаток до React необхідно додати бібліотеку для управлінням стану.

Найпопулярніша бібліотека для управління станом додатка на сьогоднішній день - це бібліотека Redux. Однак Redux не найкращий вибір для цього додатка, як це може здатися на перший погляд. Redux більше наслідує функціональний підхід до програмування, вимагає незмінної, імутабільной структури даних, а також дає повний контроль над потоком даних, через що доводиться писати багато, так званого "бойлерплейт" коду. Тому на альтернативу бібліотеки Redux була обрана бібліотека Mobx.

Mobx, на віміну від Redux, пропагує використання класів, які ближче до традиційного ООП, а також велика частина механіки управління стану реалізована всередині бібліотеки, що дозволяє зробити код більш оптимізованим і швидким в порівнянні з Redux, а об'єктно орієнтований підхід дозволить з легкістю описувати успадкування і залежність класів, які описують механіку роботи елеваторного підприємства.

Бібліотека MobX складається усього з 5 методів, для їх виклику використовують декоратори:

- @observer класи - це класи React, які спостерігають за зміною стану класів МобХ.
- @observable змінні – це змінні за якими спостерігає МобХ. Якщо у класі @observer компілятор знаходить використання @observable змінної, то @observable зміна збільшує свою кількість підписантів на одиницю і в разі зміни свого значення - відправляє до @observer сигнал на оновлення.
- @action - спонукає зміни в @observer при зміні значення @observable змінних.
- Autorun() – виконується після зміни @observer змінної.

- @computed – є аналогом @observer, але натомість являє собою стандартну get функцію js, що має власний кеш.

Приклад створеного класу для MobX зображенний на рис.3.2

```
import { observable, action,autorun} from 'mobx';

export default class NewsStore {
    @observable name = 'firstName';
    @observable sname = 'secondName';

    @action setName(name='', sname=''){
        this.name = name;
        this.sname = name;
    }

    @computed get getFullName(){
        return this.name + this.sname;
    }

}

autorun(()=>{console.log('name was changed')});
```

Рисунок 3.2 - Приклад створеного класу Mobx

3.3. Обґрунтування вибору технології NodeJs та MongoDB.

Node.js - програмна платформа, яка здійснює трансляцію JavaScript в машинний код і перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення з своїм середовищем розробки. Node.js додає можливість підключати зовнішні бібліотеки, написані на різних мовах, JavaScript взаємодіяти з пристроями введення-виведення через свій API. В основному Node.js застосовується на сервері, але є можливість розробляти на Node.js і десктопні віконні додатки використовуючи electron.

Однією з основних особливостей Node.js є швидкість. JavaScript-код, що виконується в середовищі Node.js, може бути в два або рази швидше, ніж код, написаний на мовах, на зразок Java або C, і на порядки швидше інтерпретованих мов на зразок Python або Ruby. Причиною є неблокіруючіх архітектура платформи, яка орієнтована на роботу з асинхронними функціями.

Також незаперечним плюсом буде використанням однієї мови як для браузерної розробки так і для серверної розробки, це полегшує завдання створення ізоморфних додатків з ssr браузерного коду.

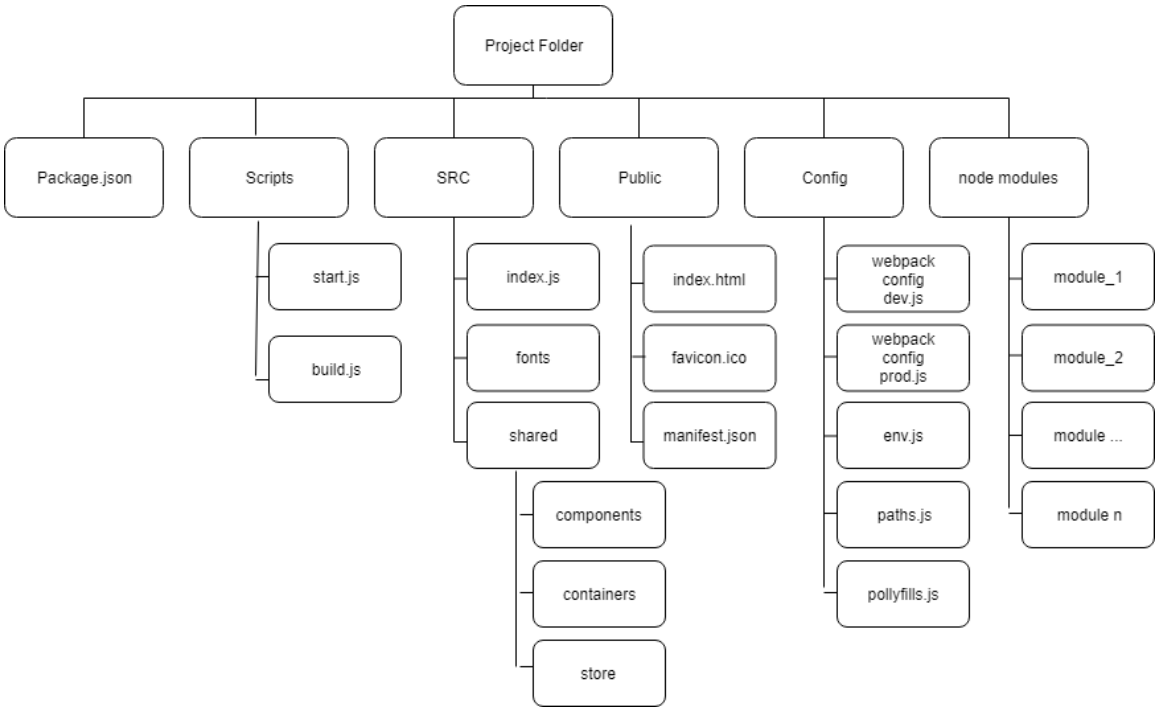
MongoDB - документоорієнтована система управління базами даних з відкритим кодом, яка не вимагає опису схеми таблиць. MongoDb класифікується як NoSQL база даних, яка використовує JSON або BSON подібні документи для зберігання записуваної інформації.

Оскільки розробляємий програмний додаток, не вимагає релятивной бази даних, а на серверы зберігаються тільки дані про користувачів і збережені дані сесії - mongoDB став кращим вибором, тому що не вимагає великої кількості часу і ресурсів для її встановлення та налаштування, на відміну від SQL баз даних.

4. ОСОБЛИВОСТІ ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ МОДЕЛЮВАННЯ ЕЛЕВАТОРНОГО ПІДПРИЄМСТВА АНАЛІЗ РОЗРОБЛЕНОГО ГРАФІЧНОГО ІНТЕРФЕЙСУ

4.1. Опис базової структури проекту.

Оскільки React є View фрейворком і більше схожий на бібліотеку, ніж на повноцінний фреймворк, для його якісного функціонування необхідна установка і визначення залежності з іншими додатковими бібліотеками, а також прописування конфігурацій виконання коду. Повна структура створеного проекту зображена на малюнку 4.1.



Риунок 4.1 - Структура створеного проекту

Для встановлення залежностей між пакетами додатку використовується Node Package Manager (NPM). npm - менеджер пакетів для мови програмування JavaScript. Цей менеджер пакету за замовчуванням використовується для виконуючого середовища в Node.js. Він складається з клієнта, командного рядка і онлайнової бази даних відкритих, платних або

приватних пакетів, яка називається реєстром npm. Доступ до реєстру здійснюється через клієнт, а доступні пакети можна переглядати і шукати за допомогою термінальної команди npm search або на веб-сайті npm.

При ініціалізації проекту з використанням пакетів Node Js в базовій директорії буде створено файл package.json. Package.json містить в собі всю службову інформацію для створеного проекту, такі як ім'я проекту, автор, поточна версія, посилання на гіт репозиторій, установка залежностей для основних пакетів і пакетів розробника, а також набори термінальних команд для їх швидкого запуску. Всі встановлені пакети потрапляють в папку node modules, яка знаходиться в кореневій папці проекту.

Для функціонування самого React були використані такі модульні залежності як react, react-dom, react-router, react-router-dom.

Пакет React містить лише функціональні можливості, необхідні для визначення компонентів React. Зазвичай він використовується разом з бібліотеками рендерінгу, наприклад React-dom для веб-додатків, або React-native для нативного середовища. Оскільки ми розробляємо веб додаток то доданий модуль рендерінгу буде бібліотека React-dom. Цей пакет служить точкою входу в DOM і засобом рендерингу для React. Він призначений для сполучення з універсальним пакетом React.

Встановлений пакет React-router забезпечує базову функціональність зі створення маршрутизації всередині програми React. Як і сам модуль react він містить тільки функціональні можливості для функціонування маршрутизації, для взаємодії з DOM деревом, аналогічно повинен бути встановлений пакет React-router-dom або react-router-native. Оскільки додаток, що розробляється є веб орієнтованим додатком, то ми встановлюємо залежність з React-router-dom пакетом.

Для транспіляції коду проекту до стандарту EcmaScript 5, а так-же перетворення JSX синтаксису до javascript коду використовується бібліотека Babel. Babel - це транспілятор, який використовується для перетворення

конструкцій, прийнятих в нових версіях стандарту ECMAScript, у вигляд, зрозумілий як сучасним, так і застарілим браузерам і іншим середах, в яких може виконуватися JavaScript. Babel, крім того, вміє перетворювати JSX-код у JavaScript, використовуючи пресет babel-preset-react.

У створенному проекті Babel виконує такі функції:

- Перетворення початкового коду
- Транспіляція коду до стандарту ES5
- Транспіляція JSX до JS
- Використання polyfill функцій

Для css файлів використовуються postcss-loader, postcss, style-loader. Як і Babel вони допомагають транспілювати код, додавати префекси і мінімізувати css файли.

Для збірки додатку ми використовуємо Webpack і Webpack Dev Server. Webpack - це статичний модульний збирач для додатків JavaScript, який допомагає упаковувати, компілювати і організовувати всі ресурси, необхідні для проекту, а також домогтися кращої організації проекту, за рахунок модульності. Він використовується разом з пакетами Babel і css-loader.

Вебпак - це дуже гнучкий інструмент і для його установки необхідно знати 4 базових поняття:

- Entry - вхід
- Output - виведення
- Loaders - завантажувачі
- Plugins – плагіни

Під Entry мається на увазі точка входу (entry point), яку вебпак буде використовувати для побудови внутрішнього графа залежностей. Після введення точки входу вебпак зможе зрозуміти, які модулі і бібліотеки зв'язуються. В результаті кожна залежність перетворюється в файли, які називаються бандли (bundles - пакети або вузли).

Output (виведення) вказує, де вебпак повинен розміщувати збірку створених бандлів.

Завантажувачі (Loaders) дозволяють вебпаку обробляти не тільки файли JavaScript, тому що сам по собі вебпак розуміє тільки JS, а трансформувати всі типи файлів в модулі, які потім можна додати в граф залежностей додатку

Webpack плагіни дозволяють кастомизувати вміст файлів згідно із заданими правилами, вони є своєрідними функціональними модулями результатом роботи яких виходить перетворений код або файл. Сам вебпак по суті є набором плагінів.

Для проекту будо створено дві конфігурації Webpack:

- конфігурація для розробки;
- конфігурація для кінцевого збирання проекту;

Скрипти для запуску збирача з відповідними настройками знаходиться в директорії scipts рис.4.1.

Конфігурація для розробки - запускає WebpackDevServer, збирає основні файли проекту, встановлює з'єднання з проксі-сервером, якщо такий заданий і перенаправляє на нього зовнішні запити, а також дозволяє моментально побачити зміни в зроблені в коді.

Конфігурація для кінцевого збирання проекту – використовується для збірки всіх файлів проекту в один великий, мінімізований бандл, готовий до завантаження на продакшн сервер. Він мінімізує js і css файли, прибирає коментарі, переноси і пробіли, транспілює код, хеширує його, збирає медійні частини в окрему директорію і інтегрує їх в кодову базу, яка буде повністю готова для візуалізації на сервері. Повний перелік конфігурацій для Webpack модулів можно знайти в папці config рис.4.1. Також там знаходиться інформація о використаних шляхах, середовищі розробки, та поліфіл функції необхідні для використання браузером.

В директорії Public (рис.4.1) знаходиться статичний вміст програми. (Index.html, manifest.json, favicon та інш.)

В директорії src (рис 4.1) знаходиться сам програмний комплекс і його складові компоненти. index.js є точкою входу для додатка і містить в собі всю маршрутизацію додатки організовану за допомогою react-router. За ним знаходяться загальні для всіх параметри стилів і шрифти, які містять в собі базові набори шрифтів і іконки позначень, shared знаходиться основна чать функціонального коду і вона розбита на три частини:

- Containers
- Components
- Store

Контейнери (Containers) – містять в собі цілісні компоненти React, рендеринг кожного з яких визначається поточною локацією додатку. Наприклад компонент <Login /> буде виводитись тільки за адресою "locallhost:3000/login", а <App /> за адресою "locallhost:3000/app". Такі компоненти складаються з більш дрібних компонентів або частин, які були розбиті спеціально з метою подальшого відтворення їх на окремих елементах сторінки. Такі компоненти знаходяться в Директорії компонент (Components) і кожен з них містить свій набір властивостей і методів, а також каскадних стилів, які дозволяли б йому реалізувати той чи Інною елемен,т інтерфейсу. Наприклад за вивід новинного блоку в контейнері <App /> рис. 4.2. відповідає компонент <News />.

Рисунок 4.2 - Візуалізація компоненту <News />

В директорії Store зберігається локальне сховище програми. Воно містить в собі основну інформацію про зернові культури, елементи елеваторного підприємства, їх вартості, коефіцієнти об'ємної ваги, терміни прибуття і відходу зернових культур, а також деякі сценарні матеріали, які

використовуються в моделюванні. Також знаходяться елементи контролю станів, які фактично є серцем програмної розробки і змушують контейнери і компоненти реагувати на їх зміни. Вони відповідають за відлік часу, підрахунок фінансів, генерування сценаріїв приходу і відходу транспорту, вхідних запитів, погодних умов і випадкових складових, як засміченість і вологість культури. Також тут знаходяться сховище для пресетів створюваної сесії, зі списком всіх елементів і вибраних користувачем і класи цих елементів, які описують їх властивості і методи, такі як сушку, чищення, об'ємна вага, поточний вагу, пропускну здатність, вологість, засміченість, стану активності, наявність сценарію і ще безліч методів по обробці цих властивостей. Усі ці класи зв’язують між собою діспетчери, які мають можливості розподіляти ресурси між всіма об’єктами, та контролювати їх взаємодію між собою.

4.1. Опис програмної розробки серверу.

Для створення серверної частини програми використовувався мову програмування NodeJs і докментно орієнтована база даних MongoDB. Повну структуру програмної частини серверу можна побачити на рис.4.3. Для створення апі додатки був використаний фреймворк express, який є програмним каркасом розробки веб-додатків для Node.js. В Директорії bin знаходиться опис конфігурації сервера, його налаштування маршрутизації, налаштування доступу до публічних дерріторіям і зв'язок з базою даних.

Директорія Routes містить в собі маршрутизацію апі і надає можливість користувачеві для авторизації, збереженню і завантаженні сесій. Node modules, аналогічно клієнтської частини програми, містить модулі залежностей описаних в package.json, , а папка build містить в собі той самий сформований клієнтською частиною програми мінімізований бандл, який віддається сервером на гет запит користувача.

Для керування базою даних серверу використовується Mongoose. Mongoose - це ORM для MongoDb зроблений під node.js.

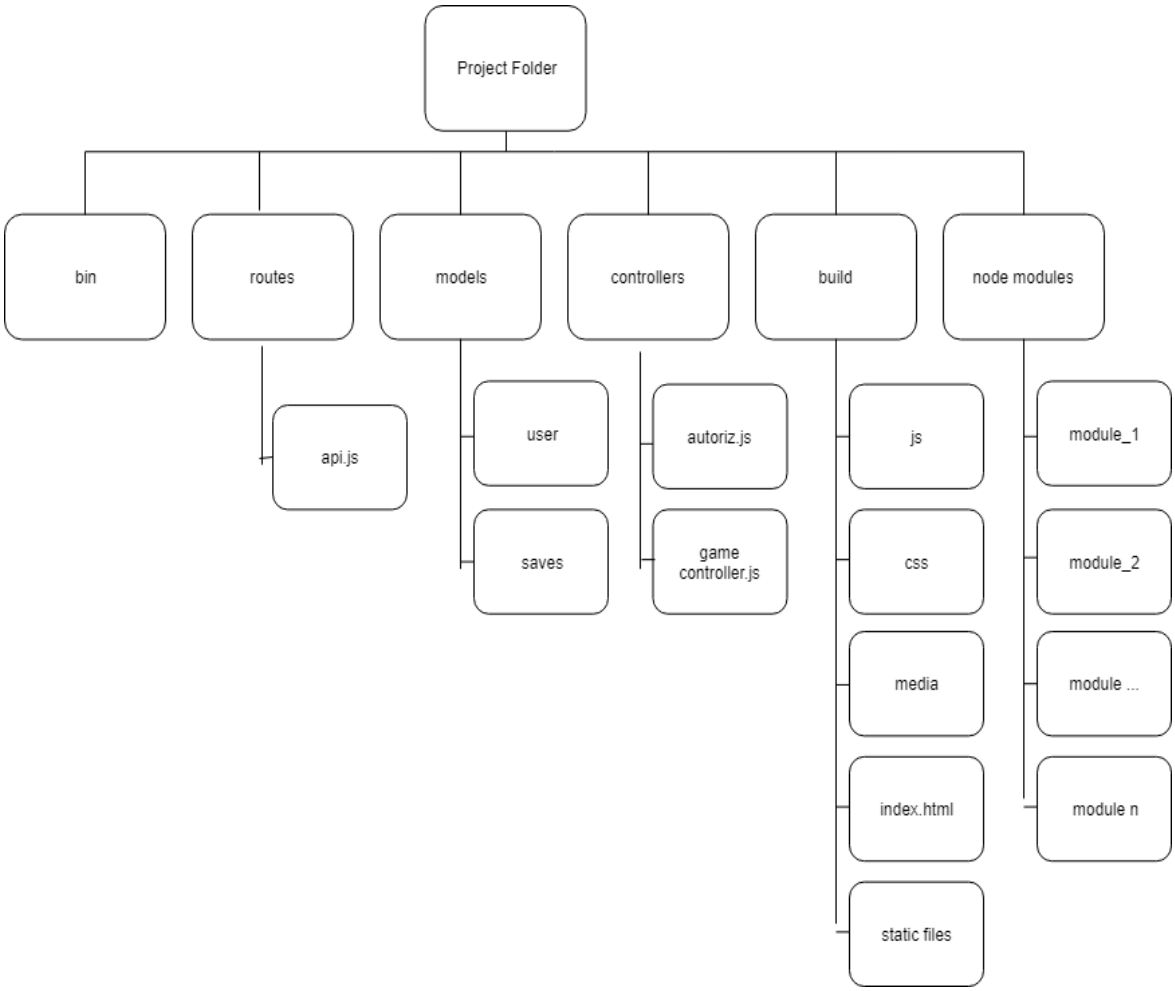


Рисунок 4.3 - Структура серверу

Оскільки MongoDB є документно-орієнтованою базою даних, то дані вона зберігає у вигляді документів json або bson. Для структуризації цих даних, прототипування зберігаємих примірників колекції і оголошення мінімальної типізації використовується Mongoose. Моделі примірників зберігається в директорії models і складається з окремої моделі, яка зберігає дані про користувачів і покажчики на його збережені сесії і модель сесій, яка зберігає самі сесії користувача, що-б їх можна було продовжити в будь-який зручний для користувача момент. Контролери містять основні методи управління і оброботки моделі mongoose, такі як збереження сесій, видалення сесії, авторизація користувача, завантаження сесії, видалення користувача і т.д.

4.2. Опис розробленого графічного інтерфейсу.

На рисунках 4.4 – 4.20 зображено графічний інтерфейс для створенної програмної розробки моделі елеваторного підприємства. Далі буде представлено опис розробленого інтерфейсу.

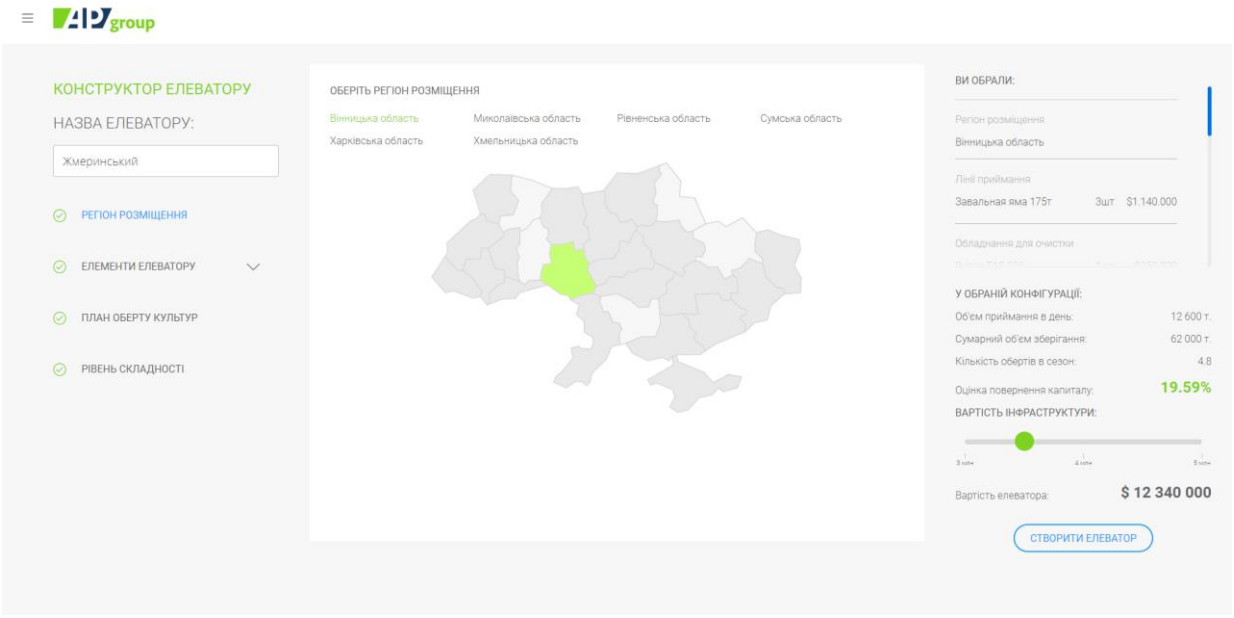


Рисунок 4.4 – Зображення вибору регіону моделювання

На рисунку 4.4 зображено інтерфейс вибору регіону моделювання. Недоступні до вибору регіони зображено сірим кольором. А вибраний регіон підсвічено зеленим кольором. Вибір регіону впливає на розподіл та структуру надходження культур, а також на погодні умови.

На рисунку 4.5 зображено інтерфейс вибору елементів приймання. На боковій панелі доступний вибір для елементів сушки, чистки, зберігання та відгрузки, який завжди буде ідентичній структурі на Рис. 4.5 Усі вибрані елементи відображаються у правій частині інтерфейсу, а кожен наступний доданий елемент додає вартість елементу до загальної вартості елеватору.

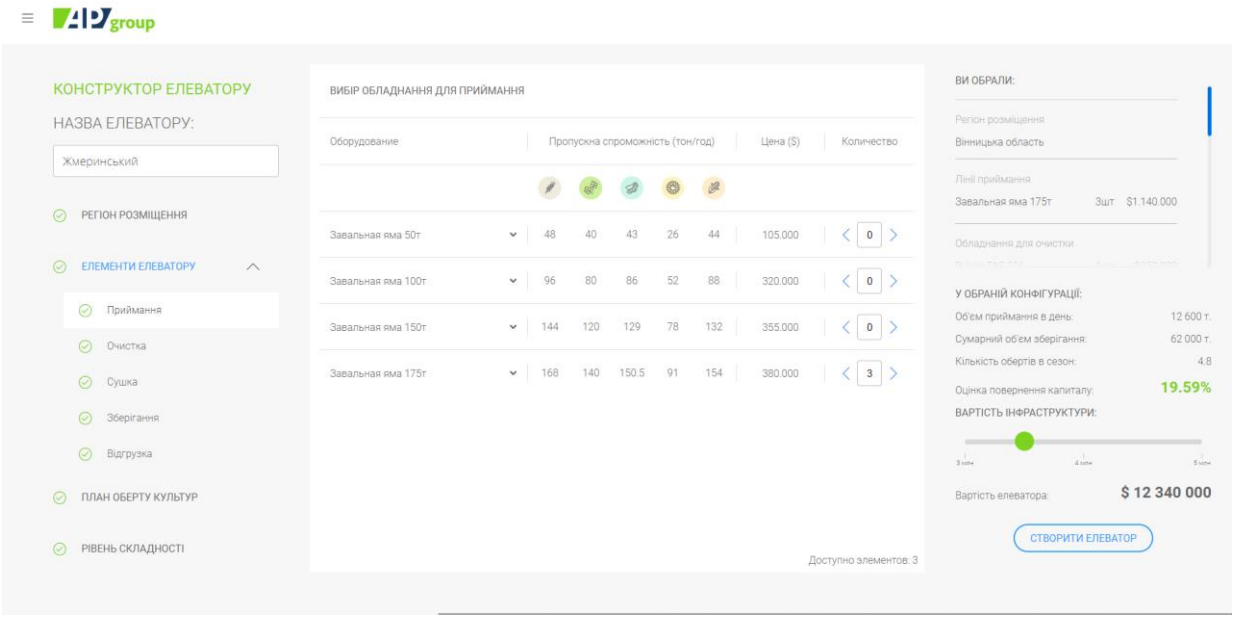


Рисунок 4.5 - Зображення вибору елементів приймання

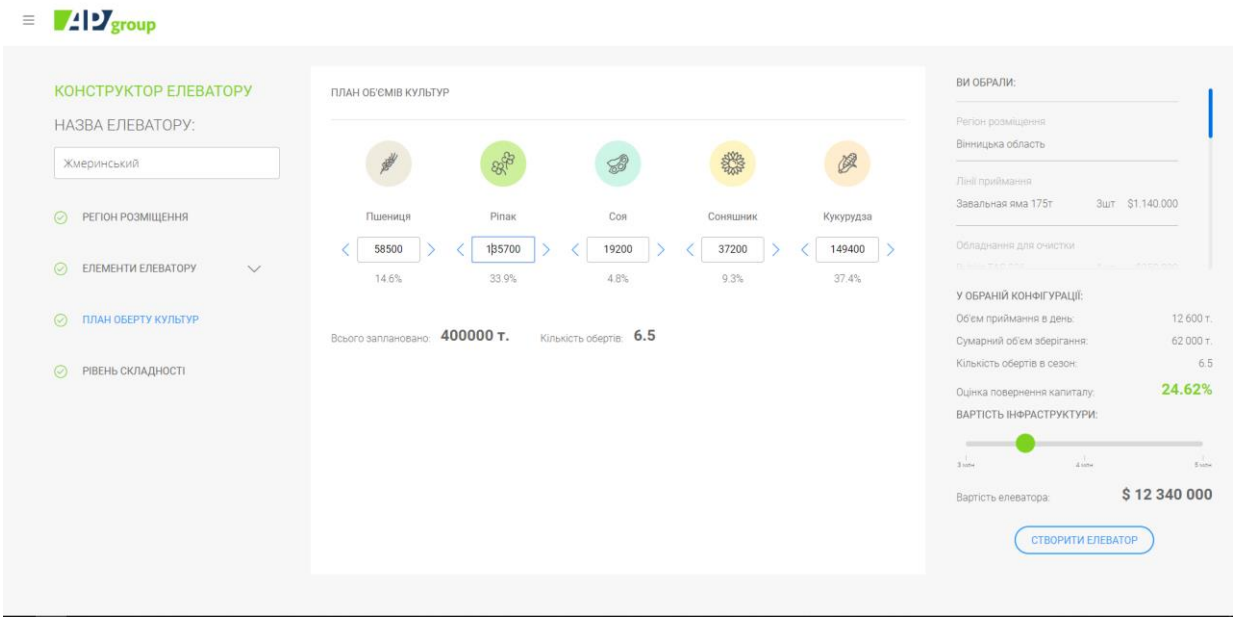


Рисунок 4.6 – Зображення налаштування розподілу культур

На рисунку 4.6 зображено інтерфейс налаштування розподілу культур. Розподіл культур впливає на попередню оцінку повернення капіталу, оборот елеватора, який розраховується з загального обраного обсягу зберігання на інтерфейсі у правій частині екрану.

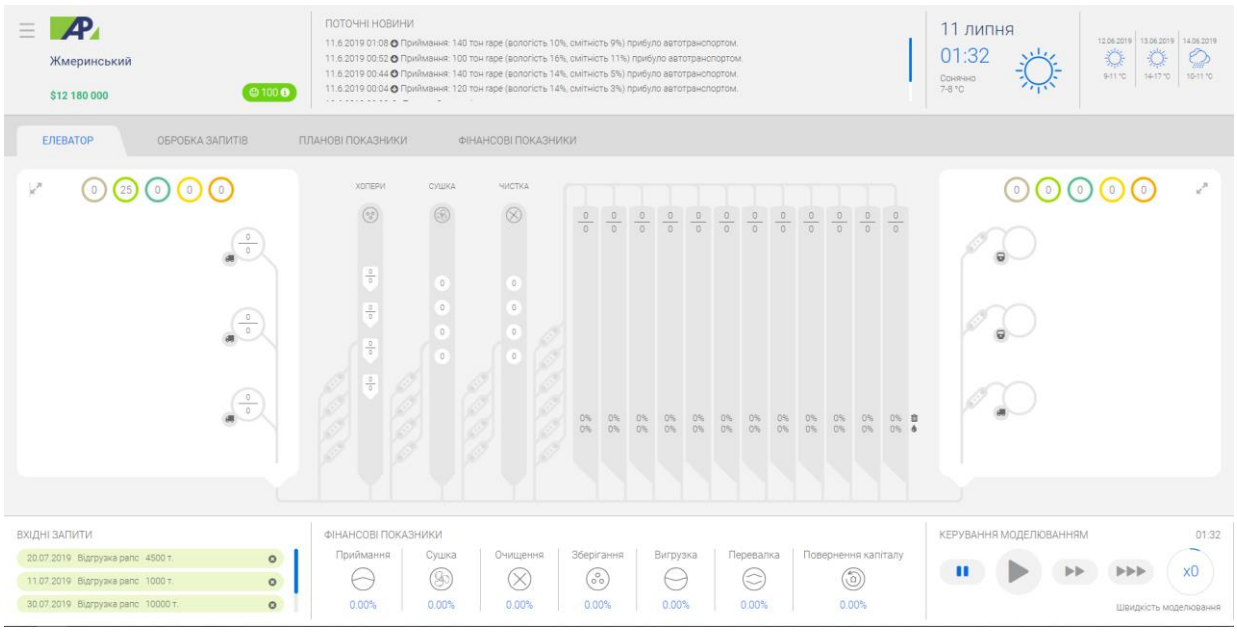


Рисунок 4.7 - Зображення умовної схеми створеного елеватора

На рисунку 4.7 зображено інтерфейс створеного елеватора у правому верхньому куті зображена назва підприємства, його загальна вартість та рейтинг співпраці контрагентами. Рейтинг співпраці з контрагентами впливає на відхилення у кількості та якості отриманого зерна на прийомному відділені та імовірність спрацювання випадкових подій. Зверху находиться новинний блок, де відображаються події, що виконуються на елеваторі. А у правому верхньому куті зображено прогноз погоди та поточну годину. У лівій частині схеми знаходиться приймальне відділення, а в правій вигрузка. По натисканню на елементи – елементу можна обрати сценарій дій. У нижний панелі знаходиться історія вхідних запитів, поточний стан фінансових показників, та система управління часом.

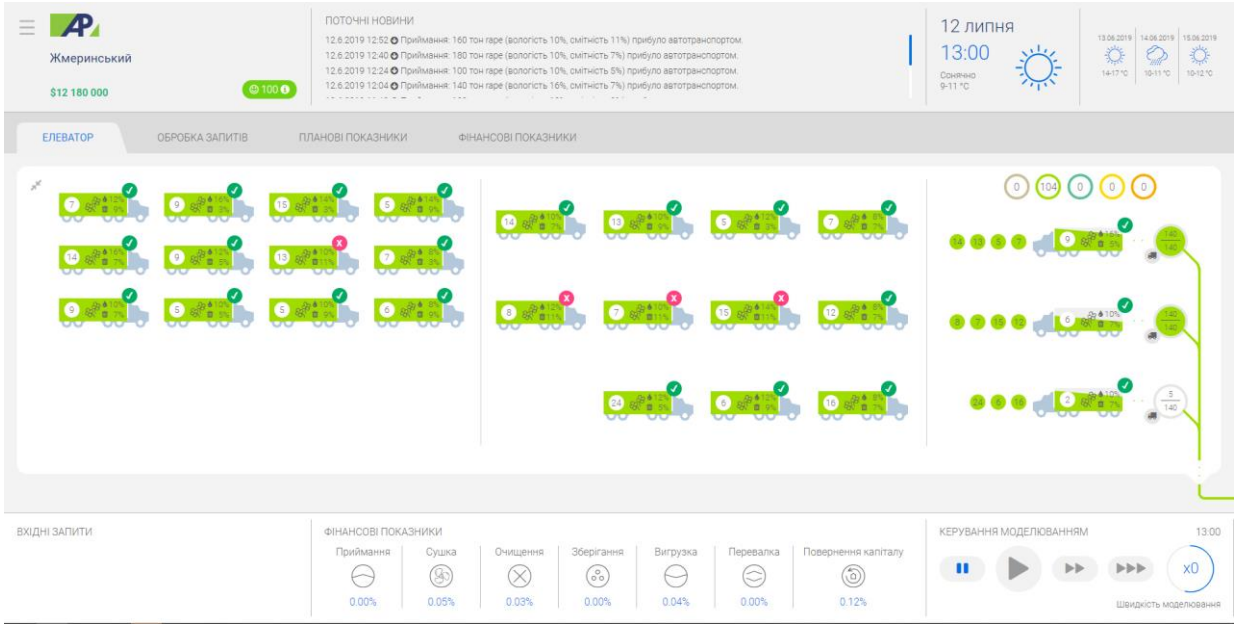


Рисунок 4.8 - Зображення умовної схеми приймального відділення

На рисунку 4.8 зображено інтерфейс приймального відділення у лівій частині інтерфейсу знаходиться стек очікування, кожний транспорт з зерновою культурою однакової вологості і засміченності групується у стек. При натисканні на машину стека, можна її затягнути у стек очікування приймання на бункері. Якщо на бункері приходить машина, то бункер починає заповнюватись зерном. Надалі бункеру можна присвоїти сценарій відгрузки.

Рисунок 4.9 - Зображення створення сценарію для бункера

На рисунку 4.9 зображено інтерфейс вибору сценарію бункера. По наведенню на елемент інтерфейсу в правій частині з’являється стислий опис обраного елемента. По натисканню на елементи в нижній частині екрану на один з елементів, його можно затягнути у загальний сценарій який надалі буде обробляти обраний елемент. Сценарій можна призначити тільки для бункерів приймання, хоперів та елементам зберігання. На рисунку 4.10 зображено результати виконання сценарію.

Рисунок 4.10 – Зображення виконання сценарію

На рисунку 4.11 зображено інтерфейс обробки запитів. В цей інтерфейс приходять запити на відватаження певної зернової культури. Кожен запит має такі параметри:

- Культура
- Дата
- Тип
- Об’єм
- Вологість
- Засміченість

Якщо прийняти запит, то на відгрузці з’явиться машини заданного типу які можуть вмістити у себе об’єм указаний в запиті. Якщо прийнятий запит не був виконаний, машини, які очікують виконання запитів зникають з відгрузки і у гравця понижається рейтинг елеватористa. Якщо запит не буде прийнятий до дати вказаної в запиті, то запит автоматично надбає відхилений статус

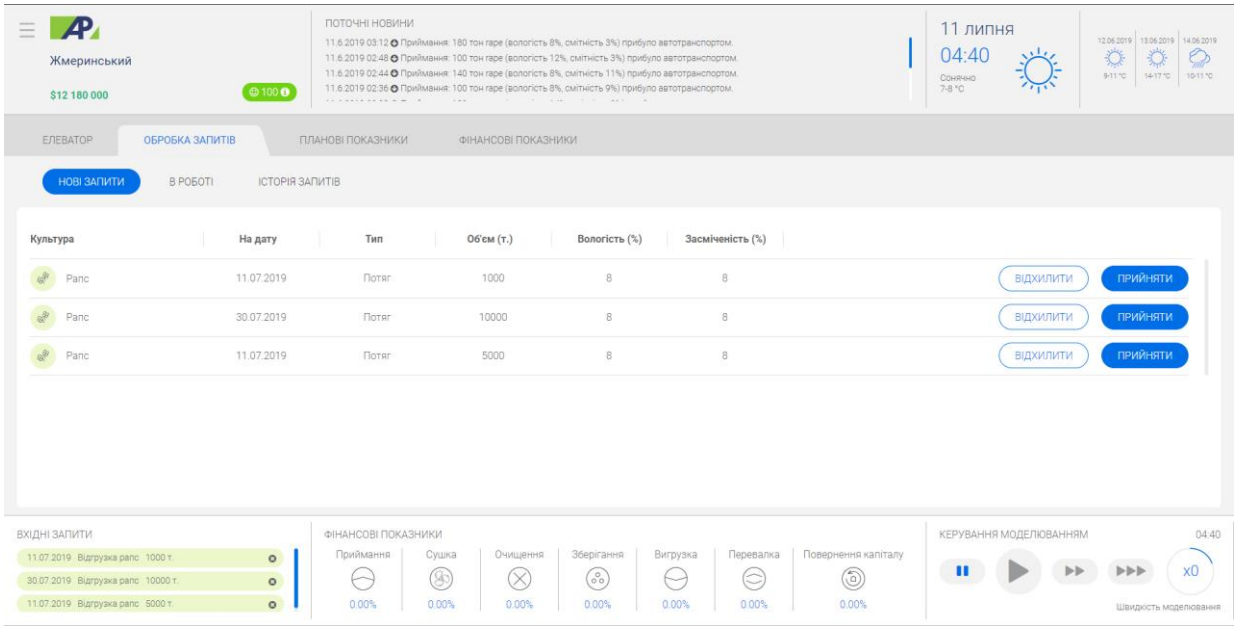


Рисунок 4.11 - Зображення інтерфейс обробки запитів

На рисунку 4.12 зображений графік прийому і відвантаження зернових культур, запланований об’єм і об’єм який залишився для відвантаження.

На рисунку 4.13 зображений аварійний елемент, який вийшов з ладу. Деякі елементи виходять з ладу самостійно, але це буває дуже рідко. Найчастіше елементи виходять з ладу через формальної помилки в обраних сценаріях. Тип помилки можна дізнатися по наведенню на забитий елемент.

Помилки бувають таких типів:

- Переповнення елементу
- Інший сценарій в елементі
- Змішання культур
- Невідповідність базісним показникам зернової культури

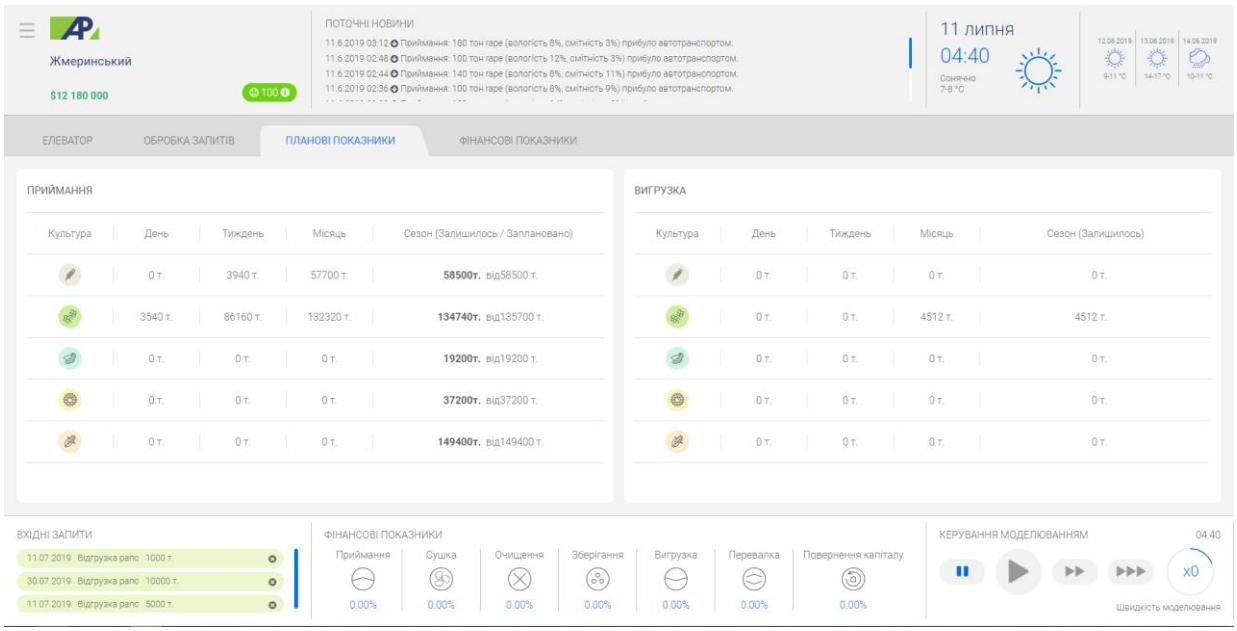


Рисунок 4.12 – Таблиці плану надходження зернових культур

Рисунок 4.13 - Зображення аварійного стану елемента

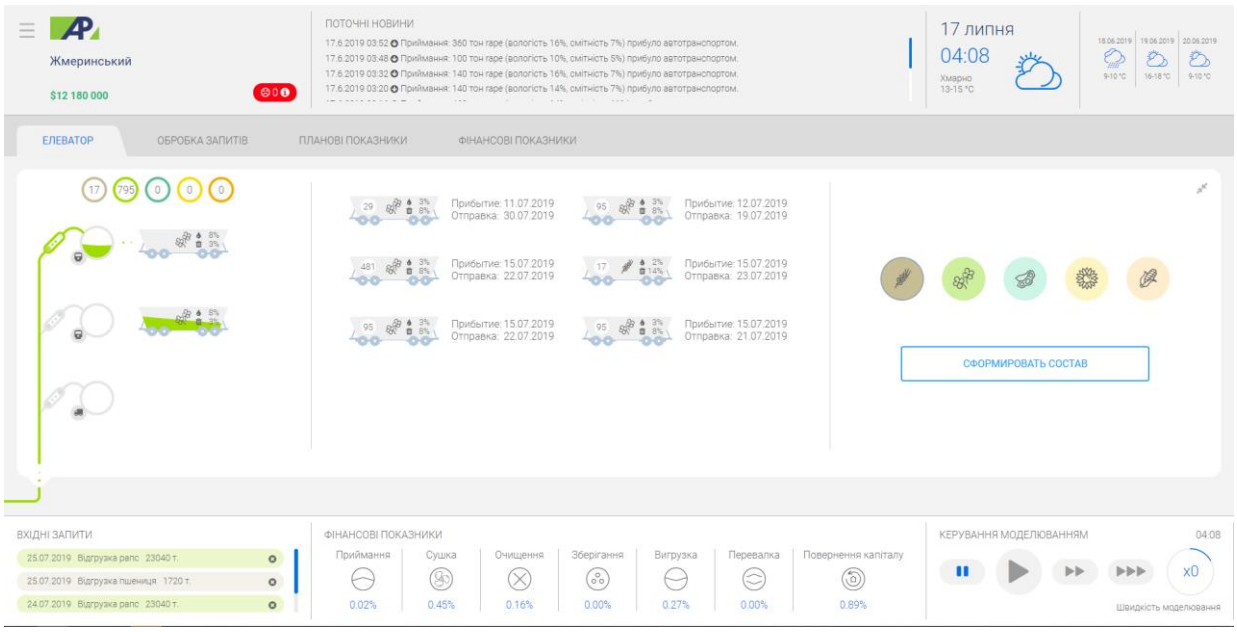


Рисунок 4.14 - Зображення інтерфейсу відгрузки

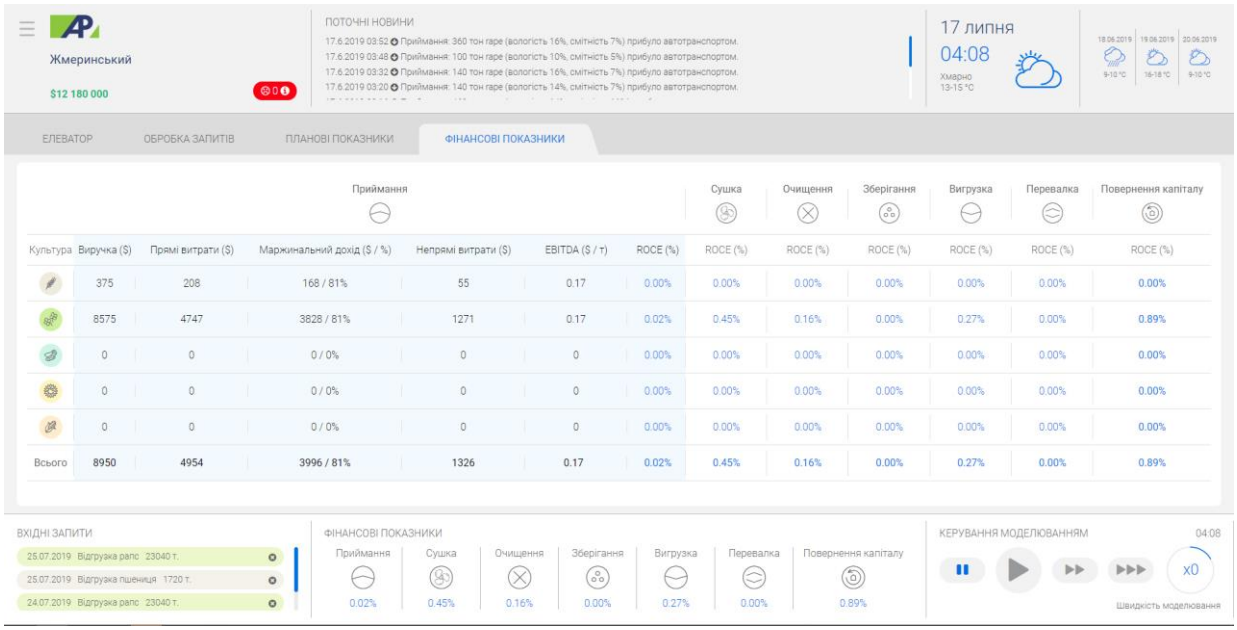


Рисунок 4.15 - Зображення фінансові показників

На рисунку 4.14 показаний умовний інтерфейс відвантаження елеватора. Якщо раніше були прийняті заявки на відвантаження зернових культур, на відвантаженні з'являється транспорт зазначеного типу, але так-же можна завжди замовити склад. У випадку замовлення залізничного склад, на наступний день з’явиться залізничний склад на 60 вагонів, який буде очікувати понад 7 днів, після чого також зникне з відгрузки.

На рисунку 4.15 можна отримати детальний фінансовий звіт по кожному елементу елеваторного підприємства. У таблиці можна отримати показники доходів, прямих і непрямих витрат, маржинальний дохід для кожної з культур, прибуток до оподаткування на тонну культури і повернення капіталу по заданому елементу.

ВИСНОВКИ

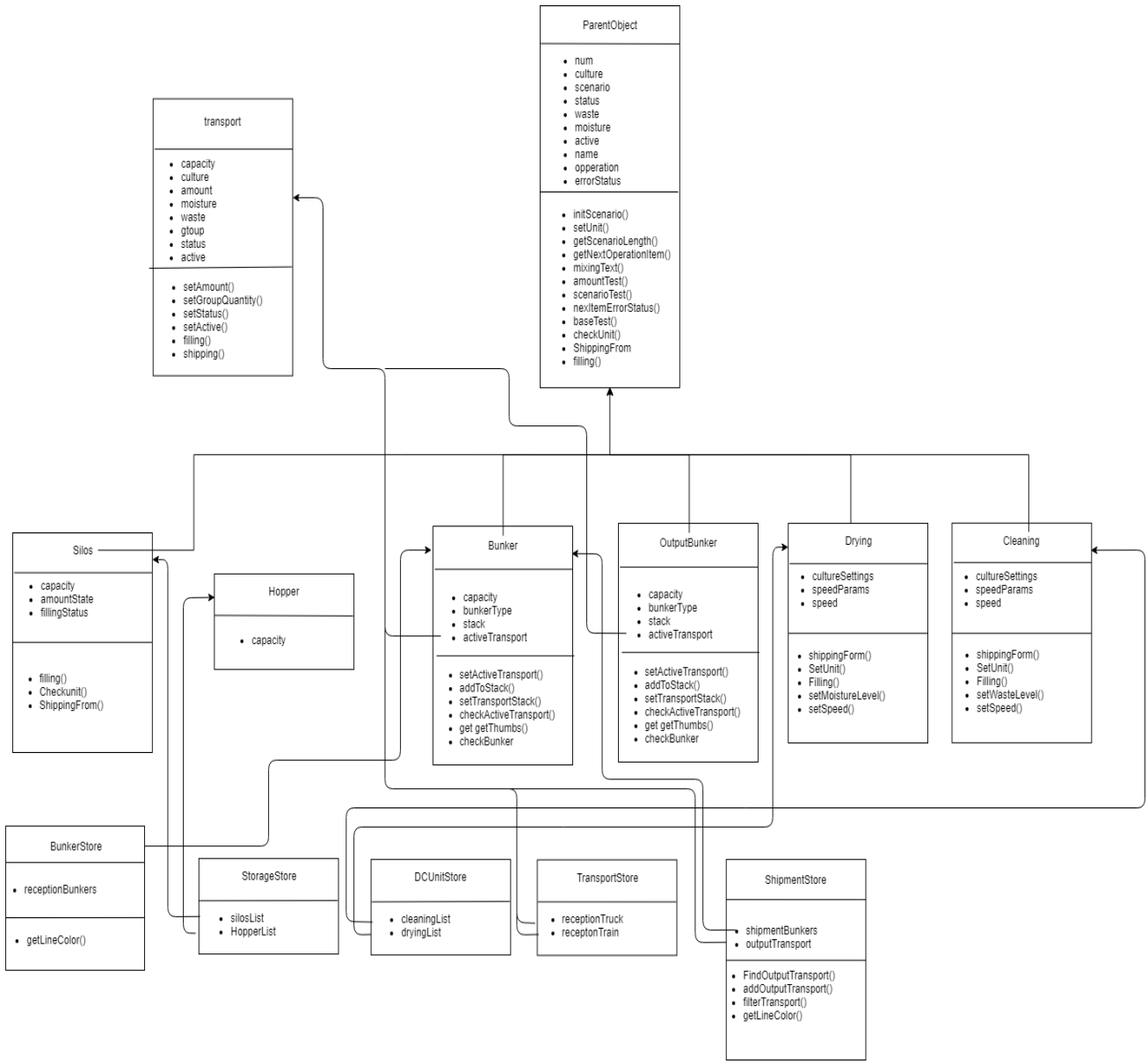
В даному дипломному проекті розроблено програмний комплекс по моделюванню елеваторного підприємства. Основною метою розробки є створення легкого багатоплатформового односторінкового веб-додатку, який дозволив би спланувати вартість витрат на будівництво інфраструктури елеватора, пропускну здатність елеватора при різній комплектації обладнання, обсяги надходження зернових культур, а також оцінити витрати і повернення капіталу за один календарний рік функціонування підприємства. Розроблений програмний комплекс дозволить навчати персонал всіх рівнів, від генеральних директорів підприємств і агрономів, до механізаторів і забудівників, включаючи підвищення знань про бізнес процес для всіх учасників виробництва з пов'язаних областей.

Модель елеваторного підприємства тестувалася 29 співробітниками компанії Агросіті і АгропросперісГруп. Сумарно було зіграно 138 сесій. Середні результати повернення капіталу по зіграним сесіям 12%, причому перший третину сесій має середній повернення капіталу 8.2, 2а - 11.4, 3я - 17.3. Повернення капіталу використовується для оцінки, як найбільш універсальний показник ефективності прийнятих рішень, тому що елеватори можуть мати різну конфігурацію. З цього можна зробити висновок що користувачі поступово покращуючи свій навик володіння інтерфейсом побудови моделі елеваторного підприємства, в тому числі підвищують свої навички прийняття рішень управління елеватором. Варто відзначити що 6 учасників впродовж усього тестування мали показник повернення капіталу більше 18%. Дані 6 учасників були директорами елеваторних підприємств побудованих за сучасними стандартами, з урахуванням всіх сучасних технологій і нововведень.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

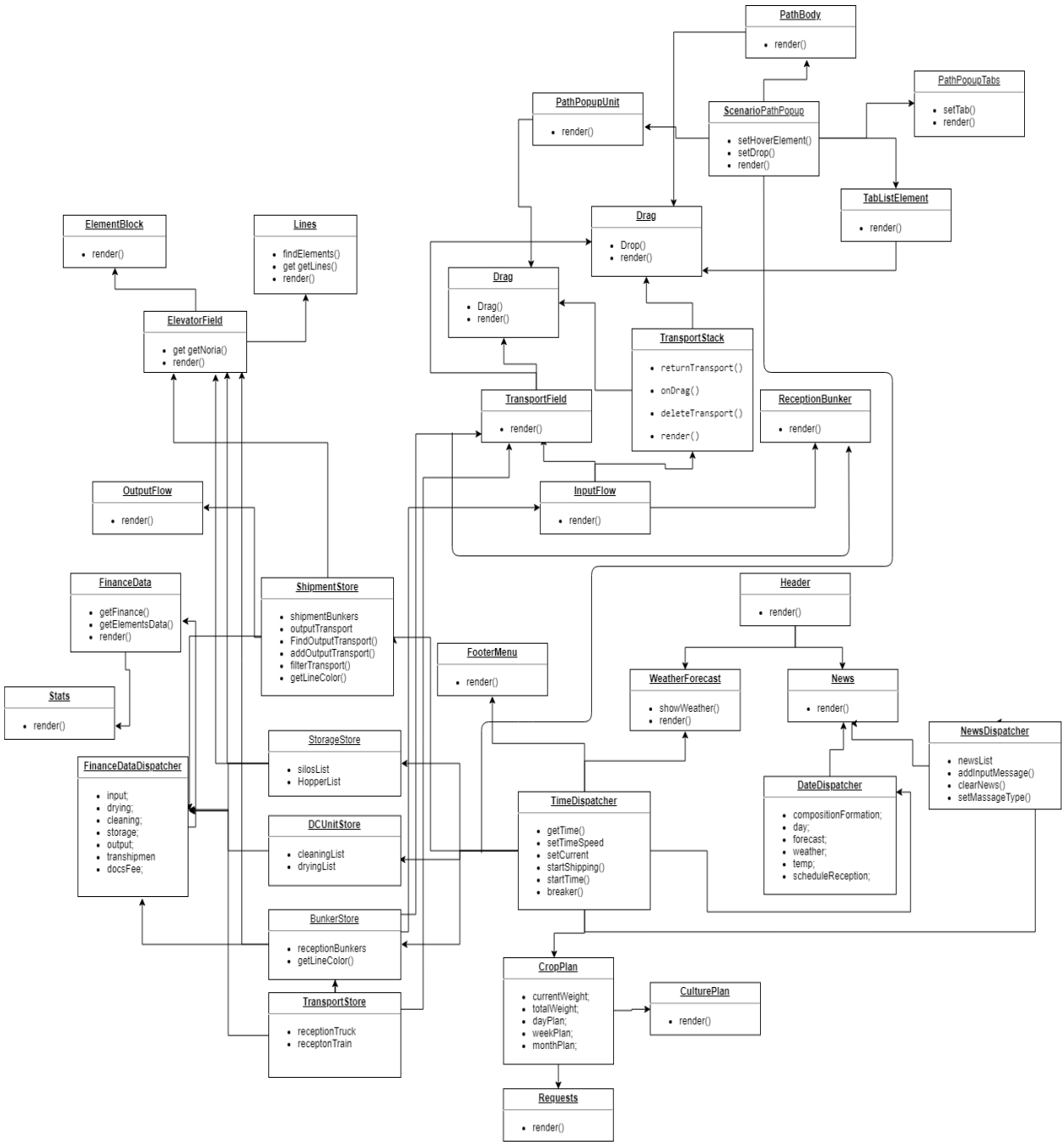
1. Волгіна, О.А. Математичне моделювання економічних процесів и систем: Навчальний посібник / О.А. Волгіна, Н.Ю. Голодна, М.М. Одіяко .. - М.: КноРус, 2012. - 200 с.
2. Документація NodeJS. – Електрон. дані (1 файл). – 2017-2019. – Режим доступу: https://nodejs.org/docs/latest-v9.x/api/. – Назва з екрану.
3. Ітан Браун. Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript = Web Development with Node and Express / Ітан Браун. - Санкт-Петербург: Пітер, 2017. - 336 с
4. Ларченко, Д.А. Інтер'єр: дизайн і комп'ютерне моделювання. / Д.А. Ларченко, А.В. Келле-. - СПб .: Пітер, 2011. - 480 с.
5. Орлова, І.В. Економіко-математичні методи і моделі: комп'ютерне моделювання: Навчальний посібник / І.В. Орлова. - М .: Вузівський підручник, НДЦ ИНФРА-М, 2013. - 389 с.
6. Рєпін, В.В. Процесний підхід до управління. Моделювання бізнес-процесів / В.В. Рєпін. - М .: Манн, Іванов і Фербер, 2013. - 544 с.
7. Струга В. П., Толкачова І. О. Імітаційне моделювання. - МГТУ ім. Баумана, 2008. - С. 697-737
8. Хемді А. Таха. Глава 18. Імітаційне моделювання // Введення в дослідження операцій = Operations Research: An Introduction. - 7-е вид. - М .: «Вільямс», 2007. - С. 697-737.
9. Gamification in Corporate Trainings on Business Analytics Tools for Agricultural Industry/ A. EIne.- Manning Publications., 2018p.
10. Introduction to MobX. – Електрон. дані (1 файл). – 2012. – Режим доступу: https://mobx.js.org/. – Назва з екрану.
11. MongoDBManual. – Електрон. дані (1 файл). – 2017-2019. – Режим доступу: https://docs.mongodb.com/manual/. – Назва з екрану.

12.React documentation. – Електрон. дані (1 файл). – 2015-2016. – Режим доступу:https://reactjs.org/docs/getting-started.html. – Назва з екрану.

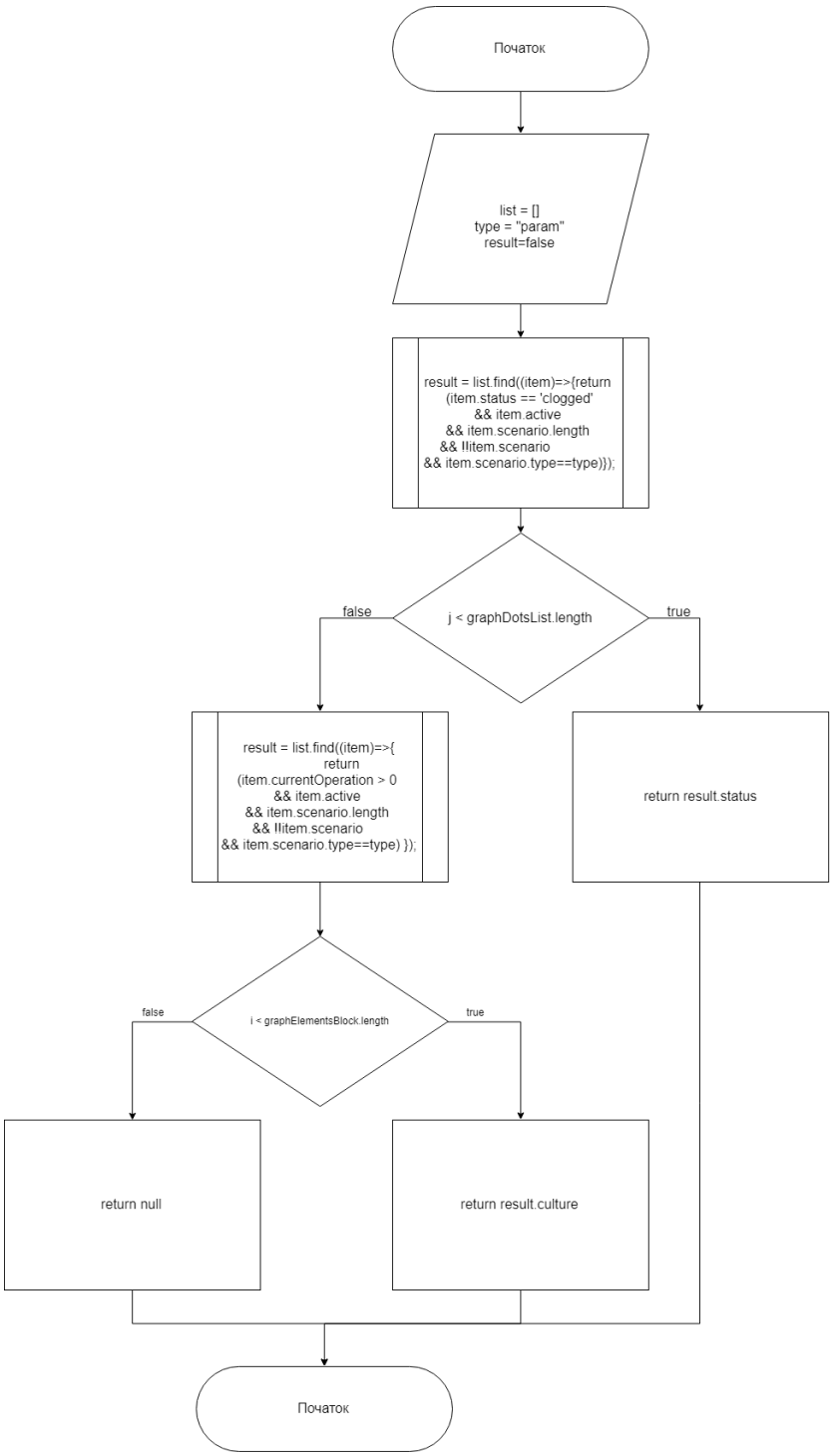


					ІАЛЦ. 454200.005Д1	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	*Схема програмного взаємозв'язку елементів локального сховища даних*	*Літ.* / *Маса* / *Масштаб*
Розроб.		*Ждан О.О.*				
Перевір.		*Клятченко*				*Арк.* 1 / *Аркушів* 1
Т. Контр.						
Н. Контр.		*Клятченко*				*КПІ ім. Ігоря Сікорського, ФПМ, КВ-53*
Затверд.		*Тарасенко*				

					ІАЛЦ. 454200.006Д2	
					Алгоритм відвантаження зернової культури з одного елемента в інший	*Літ.* / *Маса* / *Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Ждан О.О.*				*Арк. 1* / *Аркушів 1*
Перевір.		*Клятченко*				
Т. Контр.						*КПІ ім. Ігоря Сікорського, ФПМ, КВ-53*
Н. Контр.		*Клятченко*				
Затверд.		*Тарасенко*				



					ІАЛЦ. 454200.007ДЗ	
Змн.	Арк.	№ докум.	Підпис	Дата	Структура взаємодії компонентів React і контролерів стану	Літ. / Маса / Масштаб
Розроб.		Ждан О.О.				
Перевір.		Клятченко				Арк. 1 / Аркушів 1
Т. Контр.						
Н. Контр.		Клятченко			КПІ ім. Ігоря Сікорського, ФПМ, КВ-53	
Затверд.		Тарасенко				



					ІАЛЦ. 454200.008Д4	
					Алгоритм заповнення матриці суміжності для шляхів між активними елементами моделі	*Літ.* / *Маса* / *Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Ждан О.О.*				*Арк. 1* / *Аркушів 1*
Перевір.		*Клятченко*				
Т. Контр.						*КПІ ім. Ігоря Сікорського, ФПМ, КВ-53*
Н. Контр.		*Клятченко*				
Затверд.		*Тарасенко*				